VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Multimediální výukový program Návrh generátoru binárních posloupností

Multimedia Tutorial Proposal Sequences of Binary Numbers

2014 Jakub Krhovják

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Jakub Krhovják**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Multimediální výukový program Návrh generátoru binárních posloupností
Multimedia Tutorial Proposal Sequences of Binary Numbers

Zásady pro vypracování:

Cílem práce je vytvoření výukového programu pro podporu předmětu Logické obvody. Bakalářská práce bude obsahovat:
1. Teoretický popis návrhu synchronních sekvenčních obvodů.
2. Příklady návrhů generátorů posloupností binárních čísel.
3. Možné varianty návrhu zapojení a jejich vzájemné porovnání.
4. Multimediální výukový program pro danou problematiku vytvořený vybranou technologií (např. Adobe Flash Platform).

Seznam doporučené odborné literatury:

DIVIŠ, Z., CHMELÍKOVÁ, Z., ZDRÁLEK, J. *Logické obvody*. 1. vydání. Ostrava: VŠB-TUO, 2005. ISBN 80-248-0829-3.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Iva Petříková, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2014 .....

Poděkování

Rád bych poděkoval Ing. Ivě Petříkové za odbornou pomoc a konzultaci při vytváření stéto bakalářské práce.

Abstrakt

Cílem této bakalářské práce je popsat jednotlivé typy sekvenčních logických obvodů, jejich vlastnosti, proč a kde se využívají. Součástí práce bude také program Binární generátor, napsaný na v jazyce Java, ve kterém budou dle výběru uživatele představeny jednotlivé zapojení se sekvenčními obvody. Binární generátor by měl být interaktivní tzn., že by měl umožnit změnu svého chování na základě zadání vstupních hodnot. Program bude založený na knihovně Swing, určené hlavně pro tvorbu GUI, která ale umožňuje vykreslování vlastních tvarů, obrazců a také tvorbu vlastních animací.

Klíčová slova: Sekvenční logicky obvod, Java, GUI, Swing.

Abstract

The aim of this work is to describe the various types of sequential logic circuits, their properties, why and where to use them. The work will also binary generator program written in Java language, in which the user-selectable presents individual involvement with sequential circuits. Binary generator should be interactive, ie., That should allow change their behavior on the basis of an assignment of input values. The program will be based on the Swing library, designed mainly for creating GUI, but that can render custom shapes, patterns and creating your own animations.

Keywords: Sequential logic circuits, Java, GUI, Swing.

Seznam použitých zkratek a symbolů

(1)	–	Invalid Condition
74H	–	Hight speed TTL
74HCT	–	Hight speed CMOS
74L	–	Low power TTL
74LS	–	Low power Schottky TTL
74S	–	Low power TTL
ALU	–	Arithmetic logic unit
CAD	–	Computer aided design
DCTL	–	Direct-Coupled-Transistor
DTL	–	Diode-Transistor-Logic
E	–	Enable input
GUI	–	Graphics user interface
JDK	–	Java Development Kit
JVM	–	Java Virtual Machine
q^t	–	Current state
q^{t+1}	–	Following state
q	–	Inner variable
R	–	Reset
RTL	–	Resistor-Transistor-Logic
S	–	Set
T	–	Clock input
TTL	–	Transistor-Transistor-Logic
UML	–	Unified Modeling Language
x	–	Input variable
y	–	Output variable

Obsah

1 Úvod **4**

2 Historie **5**

3 Sekvenční obvody **6**
3.1 Obvod typu D latch 7
3.2 Obvod typu D flip-flop 7
3.3 Obvod J-K flip-flop 8
3.4 Posuvné registry 9
3.4.1 4-bitový posuvný registr SIPO 10
3.4.2 4-bitový posuvný registr SISO 11
3.4.3 4-bitový posuvný registr PISO 12
3.4.4 4-bitový posuvný registr PIPO 12

4 Návrh sekvenčních obvodů **14**
4.1 Generátor osmi pětibitových čísel 14
4.1.1 Generátor osmi pětibitových čísel s dekodérem 15
4.1.2 Generátor osmi pětibitových čísel bez dekodéru 19
4.1.3 Zhodnocení 23
4.2 Návrh generátoru binární posloupnosti 23
4.2.1 Generátor binární bitové posloupnosti s obvody D a J-K 23
4.2.2 Generátor binární posloupnosti s posuvným registrem 28
4.2.3 Porovnání mezi řešení s registrem a s obvody D a J-K 30
4.3 Elektronický zámek s trojcifernou kombinací čísel 0 až 9 31

5 Java a knihovna Swing **36**

6 Binární generátor **37**
6.1 Možnosti Binárního generátoru 37
6.1.1 Generátory čísel 38
6.1.2 Generátory bitové posloupnosti 39
6.1.3 Elektronický trojciferny zámek 39
6.2 Princip Binárního generátoru 40

7 Reference **43**

Seznam tabulek

1 Pravdivostní tabulka klopného obvodu typu D latch 7
2 Pravdivostní tabulka obvodu typu D flip-flop 8
3 Pravdivostní tabulka obvodu J-K flip-flop 9
4 Tabulka zobrazující postup log. 1 registrem 11
5 Tabulka přechodů pro řešení s dekodérem 15
6 Tabulka přechodů pro řešení bez dekodéru 19
7 Tabulka přechodů pro obě řešení s obvody D i J-K 24
8 Pravdivostní tabulka budící funkce Q sériového vstupu 29
9 Zakódování stavů a výstupní proměnná 32
10 Tabulka přechodů el. zámku 32

Seznam obrázků

1 Vzhled integrovaného obvodu 74HCxx . . . 5
2 Obecné znázornění sekvenčního logického obvodu . . . 6
3 Obvod typu D latch . . . 7
4 Obvod typu D flip-flop . . . 8
5 Obvod typu J-K lip-flop . . . 9
6 Obecný diagram posuvného registru . . . 10
7 Diagram posuvného registru SIPO . . . 11
8 Diagram posuvného registru SISO . . . 12
9 Diagram posuvného registru PISO . . . 12
10 Diagram posuvného registru PIPO . . . 13
11 Orientovaný graf generátoru čísel . . . 14
12 Karnaughovy mapy pro budící funkce D_2 až D_0 . . . 15
13 Karnaughovy mapy pro budící funkce JK_2 až JK_0 . . . 16
14 Karnaughovy mapy pro výstupní funkce y_4 až y_0 . . . 16
15 Schéma zapojení generátoru čísel s dekodérem s obvody D . . . 18
16 Schéma zapojení generátoru čísel s dekodérem s obvody J-K . . . 18
17 Karnaughovy mapy generátoru čísel pro budící funkce D_3 a D_2 . . . 19
18 Karnaughovy mapy generátoru čísel pro budící funkce D_1 a D_0 . . . 20
19 Karnaughovy mapy generátoru čísel pro budící funkce JK_3 až JK_0 . . . 20
20 Schéma zapojení generátoru čísel bez dekodéru s obvody D . . . 22
21 Schéma zapojení generátoru čísel bez dekodéru s obvody J-K . . . 22
22 Orientovaný graf generátoru 16-bit. poslopnosti s obvody D a J-K . . . 23
23 Karnaughovy mapy generátoru bit. posloupnosti pro budící funkce D . . . 24
24 Karnaughovy mapy generátoru bit. posloupnosti pro budící funkce J-K . . . 25
25 Karnaughova mapa pro dekodér generátoru bit. posloupnosti . . . 25
26 Schéma zapojení generátoru bit. posloupnosti s obvody D . . . 27
27 Schéma zapojení generátoru bit. posloupnosti s obvody J-K . . . 27
28 Rozdělení binární posloupnosti do pětibitových čísel . . . 28
29 Orientovaný graf pro binární posloupnost n = 5 . . . 28
30 Karnaughova mapa pro budící funkci Q sériového vstupu . . . 29
31 Realizace zapojení generátoru 16-bit. posloupnosti s posuvným registrem 30
32 Orientovaný graf el.zámku . . . 31
33 Karnaughova mapa pro budící funkci D_1 . . . 33
34 Karnaughova mapa pro budící funkci D_0 . . . 33
35 Karnaughova mapa pro výstupní funkci y . . . 33
36 Realizace elektronického zámku . . . 34
37 Úvodni obrazovka Binárního generátoru . . . 37
38 Generátor čísel s obvody D bez dekodéru . . . 38
39 Generátor bitové posloupnosti v provedení s posuvným regisrem . . . 39
40 Elektronický trojciferný zámek . . . 40
41 Zjednodušený UML diagram Binárního generátoru . . . 40
42 Implementace metody paintComponent . . . 41

1 Úvod

Současná technicky vyspělá civilizace spoléhá na nejrůznější druhy důmyslných elektronických zařízení, která člověku pomáhají v každodenní činnosti. Asi nejvíce využití najde výkonný, procesorem řízený počítač, který lze využít k nejrůznějším algoritmicky náročným operacím. Procesor počítače je tvořen miliony kombinačních logických obvodů, které tvoří například ALU, ale je sestaven také s mnoha sekvenčních logických obvodů. Sekvenční obvody se využívají při návrhu registrů a čítačů/časovačů procesoru. Vyrábí se také jako integrované obvody v plastových pouzdrech pro méně náročné aplikace.

Tato práce, která si klade za cíl vysvětlit principy fungování sekvenčních logických obvodů, na kterých staví současná výpočetní technika, je rozdělena do dvou hlavních celků, z nichž se první část zaměřuje na popis vlastností a chování sekvenčních logických obvodů. Druhá část pak popisuje program, který názorně demonstruje princip činnosti jednotlivých obvodů popsaných podrobně v první části této práce.

Kapitola 2 je věnována stručné historii logických obvodů. Na tuto kapitolu navazuje kapitola 3, objasňující co jsou to sekvenční obvody a také jsou v ní uvedeny typy obvodů, které ve své práci využívam. V kapitole 4 jsou představena jednotlivá zapojení se sekvenčními obvody, jak se postupuje při jejich návrhu a realizaci. Kapitola 5 popisuje technologie a prostředky, jež byly použity při psaní programu Binární generátor. V kapitole 6 je popsán samotný program, jeho možnosti a principy na kterých pracuje .

2 Historie

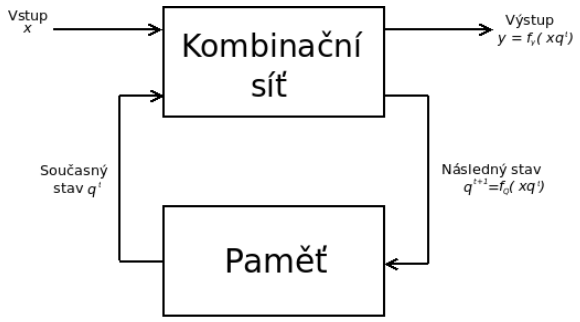
Od vnalezení tranzistoru v 50. letech minulého století, se vývoj elektroniky začal ubírat velmi rychlým tempem kupředu. Z počátku byla logická hradla realizována pomocí diskrétních součástek, což nebylo moc praktické z hlediska rozměrů a ceny. K většímu rozšíření logických obvodů došlo v 60. letech díky technologii výroby integrovaných obvodů. Ta umožnila výrobu levných, spolehlivých číslicový stavebnic s možností kaskádových zapojení. Integrované obvody prošly vývojem různých konstrukcí od nejjednodušších tranzistorových hradel DCTL a RTL přes diodové transistorové logiky DTL až po TTL hradla[6] . V roce 1965 byla uvedena řada TTL 7400, která přinesla výrazné zvýšení rychlosti, snížení spotřeby a také jejich velké rozšíření. Tuto řadu lze v různých modifikacích najít v obchodech dodnes. Např. 74L, 74H, 74S, 74LS atd. Přes všechny vylepšení TTL Obvodů, byla jejich spotřeba pro některé aplikace stále příliš vysoká a proto vyla v 70 letech vynalezena řada CMOS 4000 využívající unipolární tranzistory. Spotřeba této řády je velmi malá okolo 10 mW na jedno hradlo, ale pomalejší než TTL 7400. Postupem času se i tato řada dočkala dalších modifikací např. Rychlá řada 74HCT00 slučitelná s obvody TTL. Dalši modifikace vedly směrem k dalšímu snížení napájecího napětí se značením LV (low voltage) nebo snížení napětí logické úrovně až k 0,8 V. V dnešní době se pro logické obvody používá řada 74xx

Obrázek 1: Vzhled integrovaného obvodu 74HCxx

3 Sekvenční obvody

Sekvenční logické obvody jsou logické obvody, které jsou schopny si zapamatovat svůj předešlý stav. Jsou realizovány pomocí hradel a ty jsou zase tvořeny diskrétními součástkami jako jsou rezistory, tranzistory a diody[1][2]. U sekvenčních logických obvodů, na rozdíl od kombinačních logických obvodů, záleží nejen na okamžité hodnotě vstupů, ale i na předcházejících posloupnostech stavů. Tyto stavy jsou uchovány pomocí paměťových prvků, realizovanými klopnými obvody, ve kterých se uchovávají předchozí stavy jako vnitřní proměnné sekvenčního obvodu. Klopné obvody jsou sestaveny z logických hradel a představují nejmenší paměťovou jednotku 1 bit.

Sekvenční logický obvod se tedy skládá ze dvou část, kombinační a paměťové. Zatímco kombinační část je tvořena jako klasický kombinační obvod s určitou logickou funkcí, tak paměťovou část tvoří kombinační obvody se zpětnou vazbou tzv. bistabilní klopné obvody. Tato zpětná vazba, která přivádí výstupní signál obvodu zpět na jeho vstup, umožňuje paměťové části uchovat předchozí stav ve formě vnitřní proměnné.



Obrázek 2: Obecné znázornění sekvenčního logického obvodu

Sekvenční logické obvody mohou být zkonstruovány jako jednoduché hranou řízené flip-flop obvody nebo komplexněji třeba jako posuvné registry, čítače nebo paměťová zařízení. Sekvenční obvody se rozdělují do tří hlavních kategorií[1].

Řízené událostmi asynchronní obvod, který změní stav jakmile je mu to povoleno.

Řízené hodinami synchronní obvod, který se synchronizuje s hodinovým signálem.

Pulzně řízené jsou kombinaci dvou výše uvedených.

Sekvenční obvody lze ještě dále rozdělit podle toho, jakým způsobem na hodinový signál reagují. Synchronní obvody se dělí na řízené úrovní(latch), nebo řízené hranou(flip-flop). Úrovní řízené obvody můžou měnit své stavy po celou dobu definované úrovně hodinového signálu. Obvody citlivé na hranu hodinového signálu, mohou měnit své stavy pouze na základě sestupné nebo náběžné příchozí hrany.

Většina sekvenčních obvodů jsou vyráběna, jako synchronní sekvenční obvody řízené hranou hodinového signálu. Je to z důvodu mnohem snazšího návrhu. Nevýhodou,

oproti asynchronním obvodům, může být větší příkon a také rušení okolního prostředí, kvůli vysokým kmitočtům řídícího signálu. Navíc maximální řídící kmitočet je omezen nejpomalejší části sekvenčního obvodu.

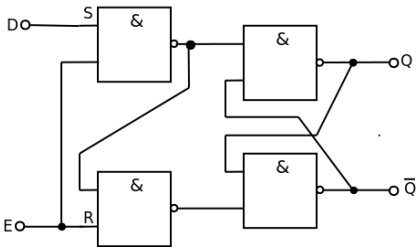
3.1 Obvod typu D latch

Obvod typu D latch je sestaven tak, že je schopen předcházet zakázaným stavům. Toho docílí tím, že má předřazen jeden logický člen NAND jako invertor[1][6], jak lze vidět na obrázku 3.Vstupní úrovně logických členů jsou vždy vzájemně invertovány, a proto nelze dosáhnout nikdy stejných hodnot na vstupech tohoto obvodu.

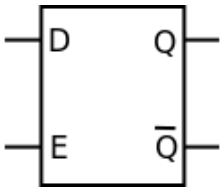
Data jsou přiváděná na vstup D, na který obvod reaguje v případě povolovacího vstupu E v logické 1. Z toho vyplývá , že obvod může měnit stav po celou dobu kdy E = 1. Chování obvodu lze snadno vyčíst z tabulky 1.

	D	E	Q^{t+1}	$\overline{Q}^{t+1}$	Chování KO
1	0	1	0	1	KO nulován
2	1	1	1	0	KO nastaven
3	X	0	Q^t	$\overline{Q}^t$	KO nemění stav

Tabulka 1: Pravdivostní tabulka klopného obvodu typu D latch



Klopný obvod typu D latch



Symbol obvodu D latch

Obrázek 3: Obvod typu D latch

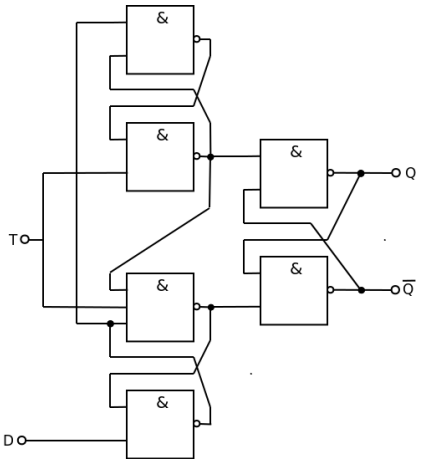
3.2 Obvod typu D flip-flop

Oproti D lath je obvod D flip-flop řízen hranou hodinového signálu. Vlastnost reagovat pouze na hranu hodinového signálu je výhodnější zejména protože reaguje přímo na příchod signálu a další reakce může nastat až s příchodem další sestupné, nebo náběžné hrany. Obvody D flip-flop mohou být řízeny jak sestupnou, tak i náběžnou hranou hodinového signálu T. Provedení klopného obvodu D řízeného náběžnou hranou, ve které se využívá tří klopných obvodů realizovaných hradly NAND, je znazorněno na

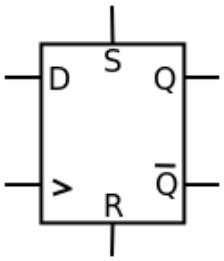
obrázku 4. Tento obvod sestává ze dvou částí. Vstupní část je tvořena dvěma klopnými obvody (vlevo), zatímco výstupní část je tvořena jedním klopným obvodem (vpravo). Chování obvodu lze opět vyčíst vyčíst z pravdivostní tabulky 2. Podrobnější popis funkcí jednotlivých hradel lze pak najit v[6].

	D	T	Q^{t+1}	$\overline{Q}^{t+1}$	Chování KO
1	0	↑	0	1	KO nulován
2	1	↑	1	0	KO nastaven
3	X	↓	Q^t	$\overline{Q}^t$	KO nemění stav
4	X	0	Q^t	$\overline{Q}^t$	KO nemění stav
5	X	1	Q^t	$\overline{Q}^t$	KO nemění stav

Tabulka 2: Pravdivostní tabulka obvodu typu D flip-flop



Obvod typu D flip-flop s hradly NAND



Symbol obvodu D latch

Obrázek 4: Obvod typu D flip-flop

3.3 Obvod J-K flip-flop

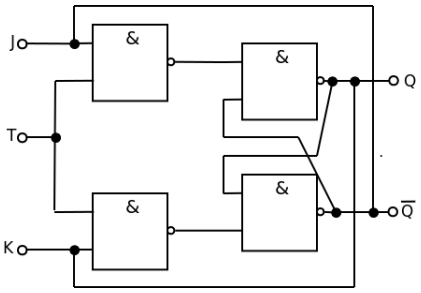
Obvod typu J-K řízený hranou je navržen tak, že pokud jsou oba vstupy J i K v logické 1, invertuje se stav obvodu. Toho se docílí tím, že má na své vstupy kříženě přivedeny hodnoty výstupů Q a $\overline{Q}$, za účasti zpětné vazby. Invertování výstupu při obou vstupech v logické 1, přináší obvodu novou kombinační funkci, čímž zase o něco rozšiřuje jeho schopnosti a zvyšuje jeho využitelnost pro potřeby návrhářů.

Obvod funguje následovně: Je-li vynulován a na výstupu J je logická 1, obvod se s příchodem aktivní hrany hodinového signálu nastaví, nezávisle na úrovni na vstupu

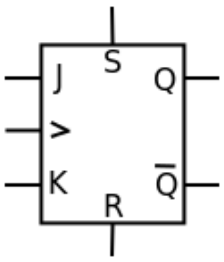
K. Je-li naopak obvod nastaven a na vstupu K je logická 1, obvod se s aktivní hranou hodinového signálu vynuluje a to nezávisle na úrovni na vstupu J.Obvody J-K jsou většinou řízeny sestupnou hranou.

	T	J	K	Q^{t+1}	$\overline{Q}^{t+1}$	Chování KO
1	X	X	X	Q^t	$\overline{Q}^t$	KO nemění stav
2	↑	0	0	Q^t	$\overline{Q}^t$	KO nemění stav
3	↑	1	0	1	0	KO nastaven
4	↑	0	1	0	1	KO nulován
5	↑	1	1	$\overline{Q}^t$	Q^t	KO invertován

Tabulka 3: Pravdivostní tabulka obvodu J-K flip-flop



Obvod typu J-K lip-flop s hradly NAND



Symbol obvodu J-K flipflop

Obrázek 5: Obvod typu J-K lip-flop

3.4 Posuvné registry

Posuvný registr je zařízení, jehož úkolem je uchovávat a posouvat informace ve formě binárních hodnot, z čehož pramení název "Posuvný registr"[9]. Posouvání hodnot je synchronizováno hodinovým signálem clk(clock). Posuvný registr je složen z několika D flip-flop obvodů, jejichž vstupy jsou navzájem kaskádově zapojeny tak, že výstup jednoho obvodu je zároveň vstup druhého obvodu atd. Data v podobě jednotlivých bitů mohou být vkládány do registru sériově, jeden po druhém a to buď z levého nebo pravého směru. Data lze vkládat, nebo číst také paralelně, všechna najednou. Název posuvného registru se odvozuje od počtu D obvodů, ze kterých je složen a tedy kolik bitů je schopen uchovat. Například 8-bitový posuvný registr, který je nejpoužívanějším z registrů, protože dokáže uchovat informaci o velikosti jeden byte.

Posuvné registry jsou tedy zejména využívány k uchovávání dat. Jsou používány v kalkulačkách, mikrokontrolérech nebo počítačích k rychlým operacím (Jsou rychlejší

než RAM paměti). Například při sčítání jsou obě proměnné načteny do registru, potom instrukci pro sčítaní sečteny a nakonec nahrány do paměti. Posuvné registry se také často využívají pro převod dat ze sériových na paralelní a naopak. Hodinové vstupy jednotlivých D flip-flop obvodů jsou paralelně spojený a řízeny taktovacím signálem clk, což zařazuje posuvné registry do synchronních obvodů.

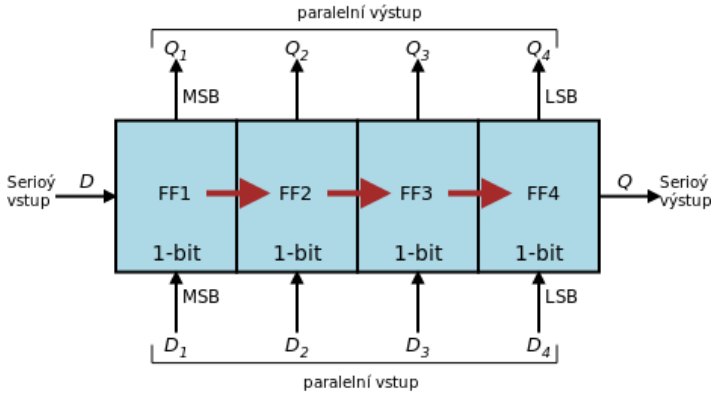
Posuvné registry jsou obvykle vybaveny R a S vstupem pro nastavení nebo resetování registru do výchozího stavu. Posuvné registry jsou rozděleny do čtyř skupin, podle způsobu jak data vstupují a procházejí celým registrem a jsou to tyto:

SIPO registr přijímá sériová data, bit za bitem, dokud nejsou dostupné z jeho výstupů v paralelní formě, (všechna data najednou).

SISO data jsou posouvány sériově ze vstupu na výstup registru, bit za bitem, zleva doprava nebo naopak.

PISO na vstupy jednotlivých D obvodů registru jsou vystavena data, všechna najednou a poté postupně vysouvána ven z registru pomocí hodinového signálu.

PIPO na vstupy jednotlivých D obvodů registru jsou vystavena data, všechna najednou a poté ve stejném hodinovém cyklu přenesena na odpovídající výstupy.



Obrázek 6: Obecný diagram posuvného registru

3.4.1 4-bitový posuvný registr SIPO

Posuvný registr SIPO funguje následovně: Na vstup R(RESET) je přivedena logická úroveň 1, čímž se registr nastaví do výchozího stavu tak, že všechny výstupy Q jsou v logické úrovni 0. Jestliže je na vstup D(DATA) obvodu FF1 přivedena logická 1, tak se při prvním hodinovém taktu přenese logická 1 na výstup Q_1, přičemž ostatní výstupy Q zůstanou v 0.

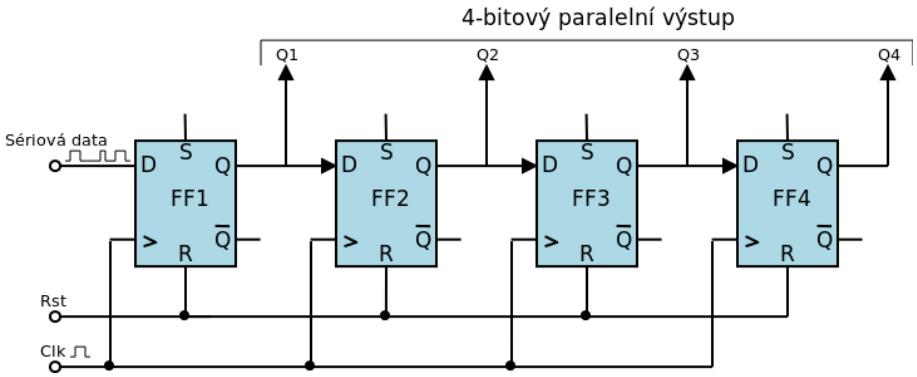
Předpokládejme, že se teď na vstup FF1 přivede hodnota 0. Druhý taktovací pulz přenese hodnotu výstup Q_1 a vstup FF2, který je 1 posune doprava na výstup Q_2. Se

třetím pulzem se logická jedna přesune na výstup obvodu FF3 a Q_3 je nyní v logické 1. Čtvrtý pulz přenese logickou 1 na poslední obvod FF4 tak, že je teď hodnota $Q_2 = 1$. Protože při posouvání, kromě prvního taktu, byl vstup D roven 0, tak je výsledná hodnota v registru 0-0-0-1. Nakonec se pátým pulzem přečtou data s registru a zároveň je celý registr vynulován přivedením logické 1 na vstup Rst.

Tímto způsobem se tedy převádějí data se sériových na paralelní. Následující pravdivostní tabulka a blokové schéma by měli pomoct si uvědomit, jak postupuje logická 1 skrze posuvný registr.

Číslo pulzu	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0

Tabulka 4: Tabulka zobrazující postup log. 1 registrem

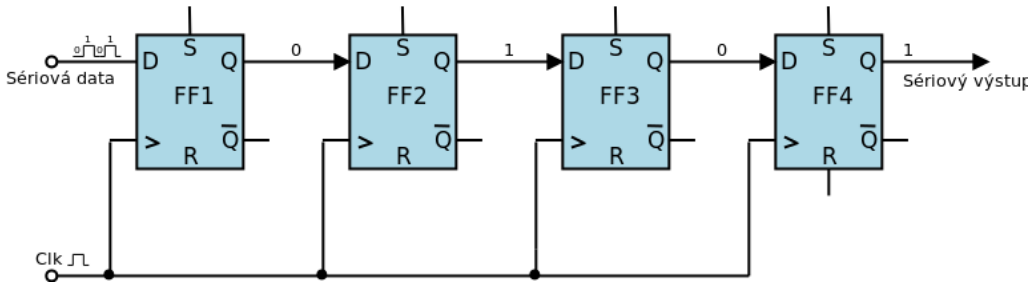


Obrázek 7: Diagram posuvného registru SIPO

3.4.2 4-bitový posuvný registr SISO

Tento posuvný registr je podobný předešlému SIPO registru, ale data z něho nejsou čtena přímo v paralelní formě z výstupů Q_1 až Q_4 . V tomto případě data tečou přímo skrze registr až na sériový výstup, jak lze vidět na obrázku 8. Protože má registr pouze jeden výstup, data jsou vysouvána z registru bit po bitu v sériovém uspořádání, odtud jméno SISO(sériově dovnitř a sériově ven).

Posuvný registr SISO je nejjednodušší z těchto čtyř registrů a jeví se jako kdyby se na něm zapojovaly jenom tři piny. Celý obvod se řídí hodinovým signálem. Tento posuvný registr se používá jako zpožďovací obvod.

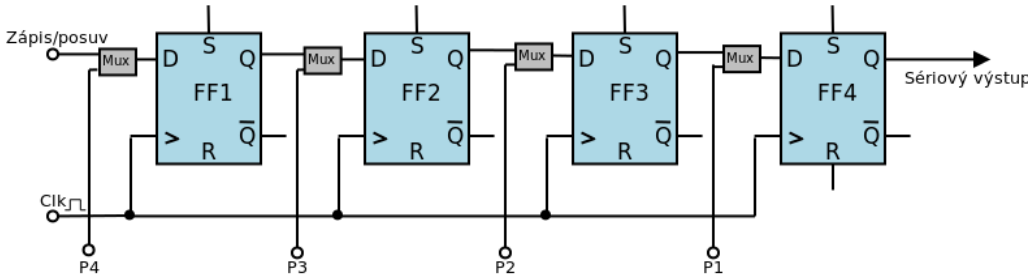


Obrázek 8: Diagram posuvného registru SISO

3.4.3 4-bitový posuvný registr PISO

Tento posuvný registr je opakem SIPO registru a plní funkci paralel-seriového převodníku dat tím, že umožňuje konverzi paralelního formátu dat na data sériové. Data jsou načtena na vstupy registru P1 až P4 zároveň. Data jsou poté čtena po jednotlivých bitech v klasickém posuvném módu, v každém hodinovém taktu signálu clk. Za povšimnutí stojí, že pro paralelní načtení dat není taktovací signál uplatněn, ten je zapotřebí pouze k posouvání dat registrem.

Tento typ registru je používán jako 8-bitový paralel-seriový převodník asynchronních signálů Tx a Rx u I/O bran mikrokontrolerů. Vyrábí se také jako stand-alone provedení např. 8-bitový 74HC166 v pozdře dip.



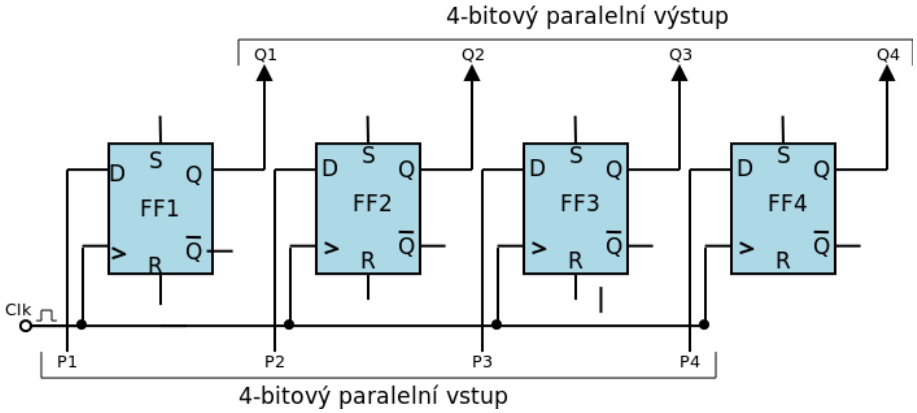
Obrázek 9: Diagram posuvného registru PISO

3.4.4 4-bitový posuvný registr PIPO

Poslední ze čtveřice je PIPO posuvný registr. Jak lze z názvu snadno odvodit, jedná se o paralel-paralel registr, který se využívá jako zpožďovač, podobně jako SISO registr popsaný výše. Data jsou nahrány najednou, stejně jako u PISO registru, na vstupy P_1 až

P_4 a poté, v jednom hodinovém pulzu, nahrány přímo na příslušné výstupní piny Q_1 až Q_4. Takže celá operace proběhne velice rychle v jediném hodinovém pulzu. Princip registru lze opět odvodit z níže uvedeného diagramu 10.

Tento typ registru se používá v konstrukci Vstupně/Výstupních bran a pracovních registrů mikrokontrolérů, kde se dočasně ukládají data k různým operacím, například ke sčítaní apod.



Obrázek 10: Diagram posuvného registru PIPO

4 Návrh sekvenčních obvodů

Při realizaci sekvenčních obvodů je možno postupovat podle následujících etap[9][[10].

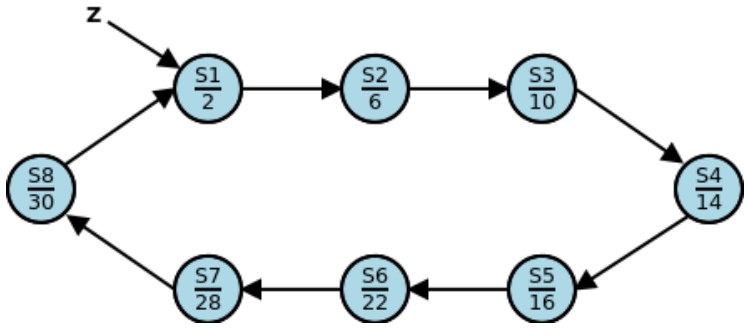
- Stanovení počtu stavových proměnných
- Zakódování stavu sekvenčního obvodu pomocí stavových proměnných.
- Stanovení následných stavů pro každý stav obvodu a každou kombinaci vstupních proměnných.
- Stanovení výstupních proměnných pro každý stav.
- Sestavení budících Booleovských funkcí.
- Sestavení výstupních Booleovských funkcí.
- Realizace výstupních a budících Booleovských funkcí.
- Realizace a nastavení počátečního stavu sekvenčního obvodu.

Návrh bude demonstrován na třech příkladech, které budou řešeny různými způsoby a s využitím rozdílných sekvenčních obvodů.

4.1 Generátor osmi pětibitových čísel

Zadaní

Na následujících stránkách se pokusím popsat jak navrhnout generátor osmi pětibitových čísel. Návrh předvedu několika způsoby s využitím jak obvodů D, tak obvodů J-K. V obou případech také vyzkouším jak ovlivní celý návrh zařazení dekodéru.



Obrázek 11: Orientovaný graf generátoru čísel

Stanovení počtu stavových proměnných a zakódování stavů

Z orientovaného grafu lze snadno vyčíst, že pro generování osmi čísel bude potřeba osmi stavů, které si zakóduji vzestupně jako čísla 0 až 7. Jelikož se jedná o pěti bitový

generátor můžu pomocí pěti bitů generovat hodnoty v rozsahu 0 až 31. Generovaná čísla jsi zvolím náhodně například 2, 6, 10, 14, 16, 22, 28 a 30.

4.1.1 Generátor osmi pětibitových čísel s dekodérem

Stanovení následných stavů pro každý stav obvodu a každou kombinaci vstupních proměnných

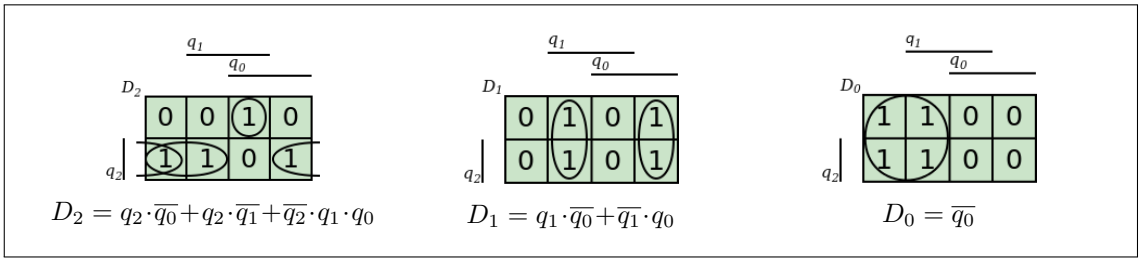
Abych navrhl jednodušší dekodér, volím při kódování stavů $q_2 = y_4, q_1 = y_3$. Pro kódování stavů volím tří vnitřní proměnné.

Zakódování stavů				Následné stavy			Výstupní proměnné				
	q_2^t	q_1^t	q_0^t	q_2^{t+1}	q_1^{t+1}	q_0^{t+1}	y_4	y_3	y_2	y_1	y_0
S_1	0	0	0	0	0	1	0	0	0	1	0
S_2	0	0	1	0	1	0	0	0	1	1	0
S_3	0	1	0	0	1	1	0	1	0	1	0
S_4	0	1	1	1	0	0	0	1	1	1	0
S_5	1	0	0	1	0	1	1	0	0	0	0
S_6	1	0	1	1	1	0	1	0	1	1	0
S_7	1	1	0	1	1	1	1	1	1	0	0
S_8	1	1	1	0	0	0	1	1	1	1	0

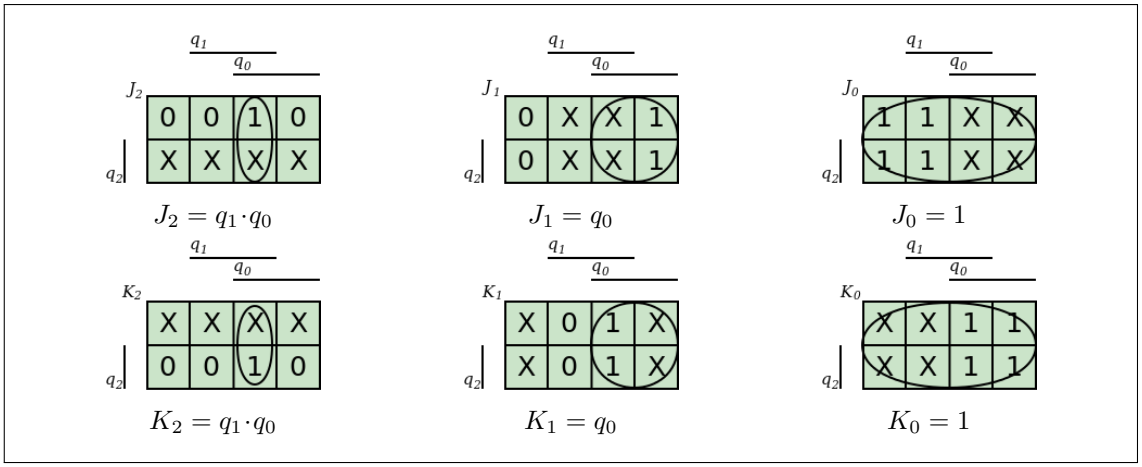
Tabulka 5: Tabulka přechodů pro řešení s dekodérem

Sestavení budících Booleovských funkcí pro obvody D a J-K

Pro sestavení budících funkcí sekvenčních obvodů využiji Karnaughovy mapy, ze kterých vyjádřím rovnice v součtovém tvaru. Budící funkce jsou zavislé na třech vnitřních proměnných q_2, q_1, q_0. Nejprve uvedu mapy pro budící funkce s obvody D a následně pro obvody J-K.



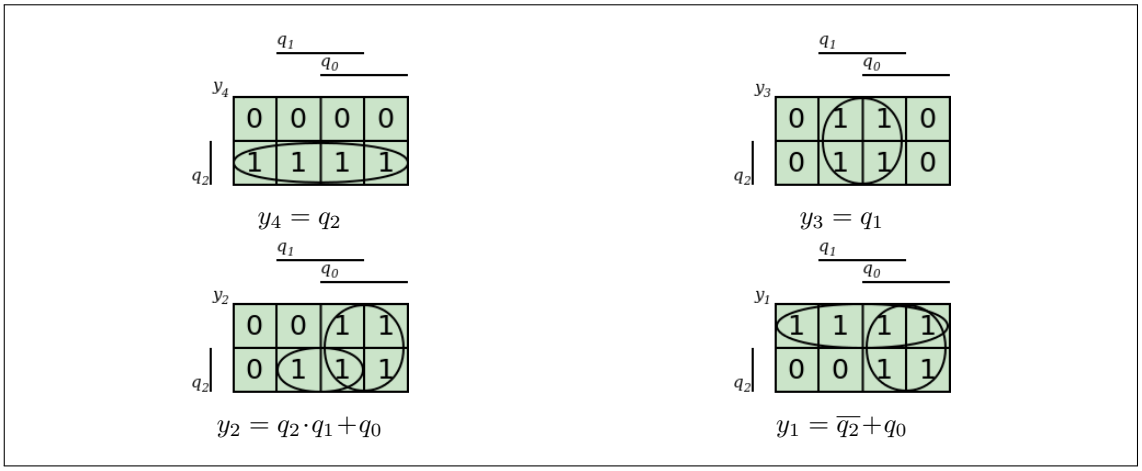
Obrázek 12: Karnaughovy mapy pro budící funkce D_2 až D_0



Obrázek 13: Karnaughovy mapy pro budící funkce JK_2 až JK_0

Sestavení výstupních Booleovských funkcí

K sestavení výstupních funkcí využiji opět Karnaughovy mapy. Vzhledem k tomu, že mnou zvolená čísla jsou všechna sudá, vystačím si pouze se čtyřmi mapami a proměné y_0 přiřadím nulu.



Obrázek 14: Karnaughovy mapy pro výstupní funkce y_4 až y_0

Sestavení rovnic pro realizaci výstupních a budících funkcí s hradly NAND

Výsledné rovnice je třeba převést do formy vhodné pro realizaci pomocí hradel NAND. Rovnice dvakrát neguji a upravím pomocí De Morganových pravidel na tvar obsahující jen součiny.

Budící funkce pro řešení s obvody D

$$D_2 = q_2 \cdot \overline{q_0} + q_2 \cdot \overline{q_1} + \overline{q_2} \cdot q_1 \cdot q_0 = \overline{\overline{q_2 \cdot \overline{q_0}} + q_2 \overline{q_1} + \overline{q_2} \cdot q_1 \cdot q_0} = \overline{\overline{q_2 \cdot \overline{q_0}} \cdot \overline{q_2 \overline{q_1}} \cdot \overline{\overline{q_2} \cdot q_1 \cdot q_0}}$$

$$D_1 = q_1 \cdot \overline{q_0} + \overline{q_1} \cdot q_0 = \overline{\overline{q_1 \cdot \overline{q_0}} + \overline{q_1} \cdot q_0} = \overline{\overline{q_1 \cdot \overline{q_0}} \cdot \overline{\overline{q_1} \cdot q_0}}$$

$$D_0 = \overline{q_0}$$

Budící funkce pro řešení s obvody J-K

$$J_2 = q_1 \cdot q_0 = \overline{\overline{q_1 \cdot q_0}} \qquad K_2 = q_1 \cdot q_0 = \overline{\overline{q_1 \cdot q_0}}$$

$$J_1 = q_0 \qquad K_1 = q_0$$

$$J_0 = 1 \qquad K_0 = 1$$

Výstupní funkce y pro obě řešení s dekodérem

$$y_4 = q_2$$

$$y_3 = q_1$$

$$y_2 = q_2 \cdot q_1 + q_0 = \overline{\overline{q_2 \cdot q_1 + q_0}} = \overline{\overline{q_2 \cdot q_1} \cdot \overline{q_0}}$$

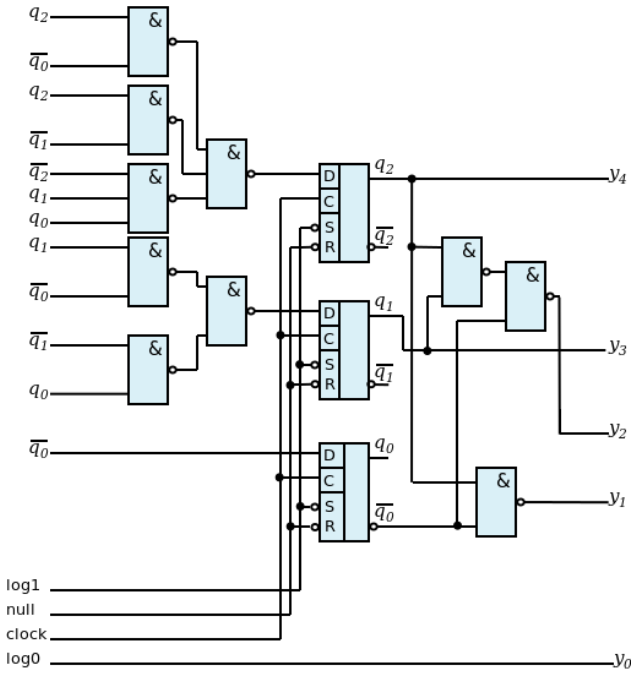
$$y_1 = \overline{q_2} \cdot q_0 = \overline{\overline{q_2} + q_0} = \overline{q_2 \cdot \overline{q_0}}$$

$$y_0 = 0$$

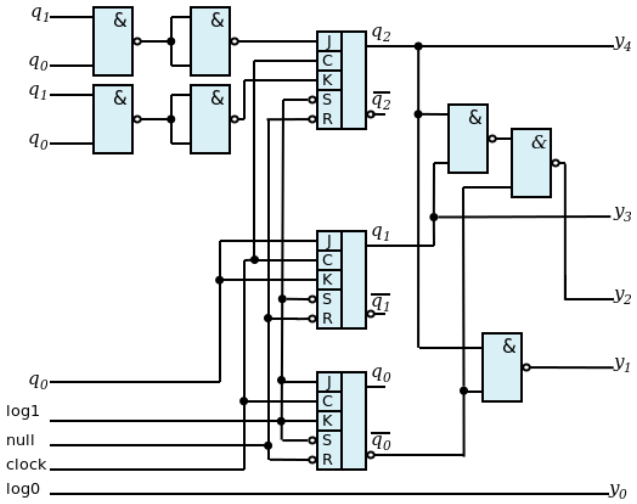
Realizace a nastavení počátečního stavu sekvenčních obvodů D a J-K

K realizaci generátoru čísel s dekodérem, použiji výsledné rovnice budících a výstupních funkcí, upravené pro použití hradel NAND. Schéma zapojení pro realizaci s obvody D je na obrázku 15, shcéma pro realizaci s obvody J-K na obrázku 16. Počáteční stav je u všech obvodů obou řešení v logické 0, a proto jsou všechny obvody na schématu vynulovány, (všechna $R = 0$ a $S = 1$).

K vytvoření schémat jsem využil program DIA, který má mnoho předdefinovaných šablon pro tvorbu různých značek a symbolů včetně logických členů. Stávající šablona pro logické členy mi však nevyhovovala, a proto jsem si vše nakreslil ručně dle svých potřeb. Tento program jsem použil k výrobě i všech ostatních schémat a také Karnaughových map.



Obrázek 15: Schéma zapojení generátoru čísel s dekodérem s obvody D



Obrázek 16: Schéma zapojení generátoru čísel s dekodérem s obvody J-K

4.1.2 Generátor osmi pětibitových čísel bez dekodéru

Stanovení následných stavů pro každý stav obvodu a každou kombinaci vstupních proměnných

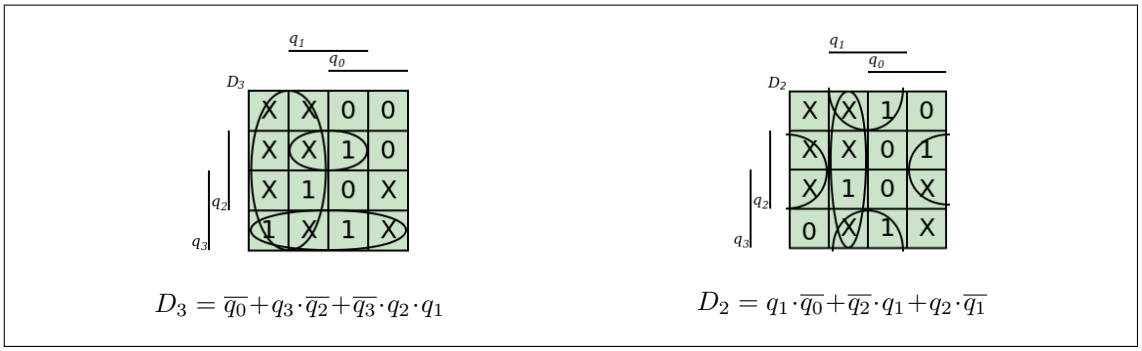
Při tomto řešení se stavy kódují tak, aby obecně platilo $q = y$. Jako u předchozích dvou řešení je výstupní proměnná y_0 trvale přivedena na zem, tedy do stavu log 0. Díky tomu si při zakódování následných stavů vystačím pouze se čtyřmi proměnnými q_0 až q_3.

Zakódování stavů					Následné stavy			
	q_3^t	q_2^t	q_1^t	q_0^t	q_3^{t+1}	q_2^{t+1}	q_1^{t+1}	q_0^{t+1}
S_1	0	0	0	1	0	0	1	1
S_2	0	0	1	1	0	1	0	1
S_3	0	1	0	1	0	1	1	1
S_4	0	1	1	1	1	0	0	0
S_5	1	0	0	0	1	0	1	1
S_6	1	0	1	1	1	1	1	0
S_7	1	1	1	0	1	1	1	1
S_8	1	1	1	1	0	0	0	1

Tabulka 6: Tabulka přechodů pro řešení bez dekodéru

Sestavení budících Booleovských funkcí pro obvody D a J-K

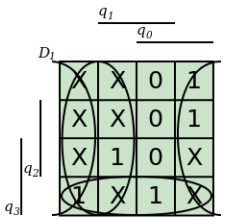
V tomto řešení jsou výsledné Karnaughovay mapy dvojnásobně velké(pro čtyři proměnné místo tří), což vyplývá z faktu, že není použit dekodér. Výsledné budící funkce budou tedy o něco složitější protože musí řešit nejen přechod do dalšího stavu, ale i to aby stavy binárně odpovídaly generovaným číslům.



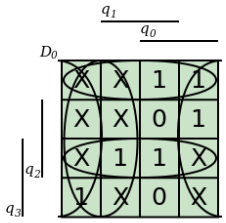
$D_3 = \overline{q_0} + q_3 \cdot \overline{q_2} + \overline{q_3} \cdot q_2 \cdot q_1$

$D_2 = q_1 \cdot \overline{q_0} + \overline{q_2} \cdot q_1 + q_2 \cdot \overline{q_1}$

Obrázek 17: Karnaughovy mapy generátoru čísel pro budící funkce D_3 a D_2

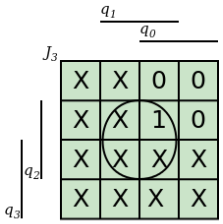


$D_1 = \overline{q_0} + q_3 \cdot \overline{q_2} + \overline{q_1}$

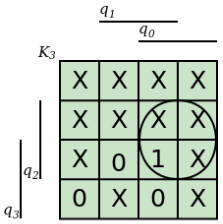


$D_0 = \overline{q_1} + \overline{q_3} \cdot \overline{q_2} + q_3 \cdot q_2$

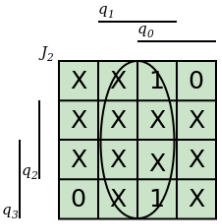
Obrázek 18: Karnaughovy mapy generátoru čísel pro budící funkce D_1 a D_0



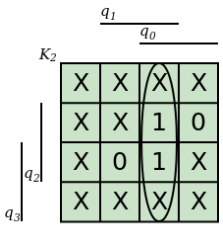
$J_3 = q_2 \cdot q_1$



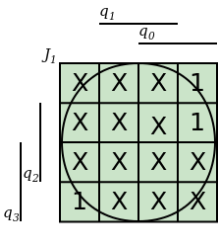
$K_3 = q_2 \cdot q_0$



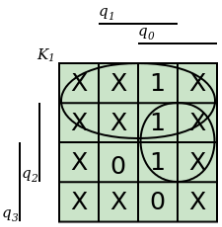
$J_2 = q_1$



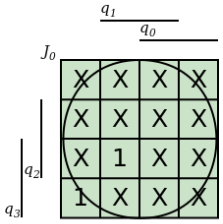
$K_2 = q_1 \cdot q_0$



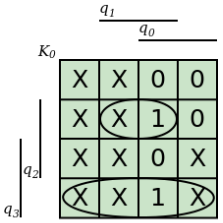
$J_1 = 1$



$K_1 = \overline{q_3} + q_2 \cdot q_0$



$J_0 = 1$



$K_0 = q_3 \cdot \overline{q_2} + \overline{q_3} \cdot q_2 \cdot q_1$

Obrázek 19: Karnaughovy mapy generátoru čísel pro budící funkce JK_3 až JK_0

Sestavení rovnic pro realizaci výstupních a budících funkcí s hradly NAND

Výsledné rovnice opět převedu do formy, která se využívá při realizaci pomocí hradel NAND.

Budící funkce pro řešení s obvody D

$$D_3 = \overline{q_0}+q_3\cdot\overline{q_2}+\overline{q_3}\cdot q_2\cdot q_1 = \overline{\overline{\overline{q_0}+q_3\cdot\overline{q_2}+\overline{q_3}\cdot q_2\cdot q_1}} = \overline{q_0\cdot\overline{q_3\cdot\overline{q_2}}\cdot\overline{\overline{q_3}\cdot q_2\cdot q_1}}$$

$$D_2 = q_1\cdot\overline{q_0}+\overline{q_2}\cdot q_1+q_2\cdot\overline{q_1} = \overline{\overline{q_1\cdot\overline{q_0}+\overline{q_2}\cdot q_1+q_2\cdot\overline{q_1}}} = \overline{\overline{q_1\cdot\overline{q_0}}\cdot\overline{\overline{q_2}\cdot q_1}\cdot\overline{q_2\cdot\overline{q_1}}}$$

$$D_1 = \overline{q_0}+q_3\cdot\overline{q_2}+\overline{q_1} = \overline{\overline{\overline{q_0}+q_3\cdot\overline{q_2}+\overline{q_1}}} = \overline{q_0\cdot\overline{q_3\cdot\overline{q_2}}\cdot q_1}$$

$$D_0 = \overline{q_1}+\overline{q_3}\cdot\overline{q_2}+q_3q_2 = \overline{\overline{\overline{q_1}+\overline{q_3}\cdot\overline{q_2}+q_3\cdot q_2}} = \overline{q_1\cdot\overline{\overline{q_3}\cdot\overline{q_2}}\cdot\overline{q_3\cdot q_2}}$$

Budící funkce pro řešení s obvody J-K

$$J_3 = q_2\cdot q_1 = \overline{\overline{q_2\cdot q_1}} \qquad K_3 = q_2\cdot q_0 = \overline{\overline{q_2\cdot q_0}}$$

$$J_2 = q_1 \qquad K_2 = q_1\cdot q_0 = \overline{\overline{q_1\cdot q_0}}$$

$$J_1 = 1 \qquad K_1 = \overline{q_3}+q_2\cdot q_0 = \overline{\overline{\overline{q_3}+q_2\cdot q_0}} = \overline{q_3\cdot\overline{q_2\cdot q_0}}$$

$$J_0 = 1 \qquad K_0 = q_3\cdot\overline{q_2}+\overline{q_3}\cdot q_2\cdot q_1 = \overline{\overline{q_3\cdot\overline{q_2}+\overline{q_3}\cdot q_2\cdot q_1}} = \overline{q_3\cdot\overline{q_2}\cdot\overline{\overline{q_3}\cdot q_2\cdot q_1}}$$

Výstupní funkce y pro obě řešení bez dekodéru

$$y_4 = q_3$$

$$y_3 = q_2$$

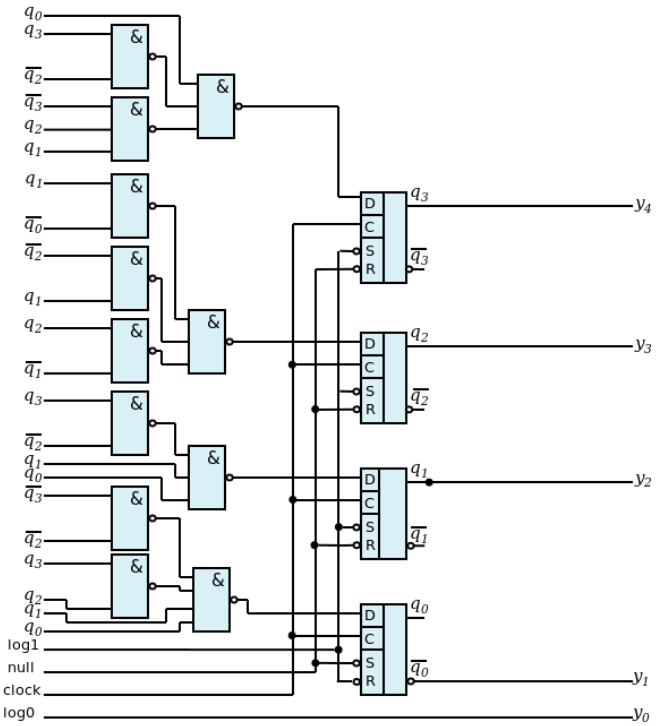
$$y_2 = q_1$$

$$y_1 = q_0$$

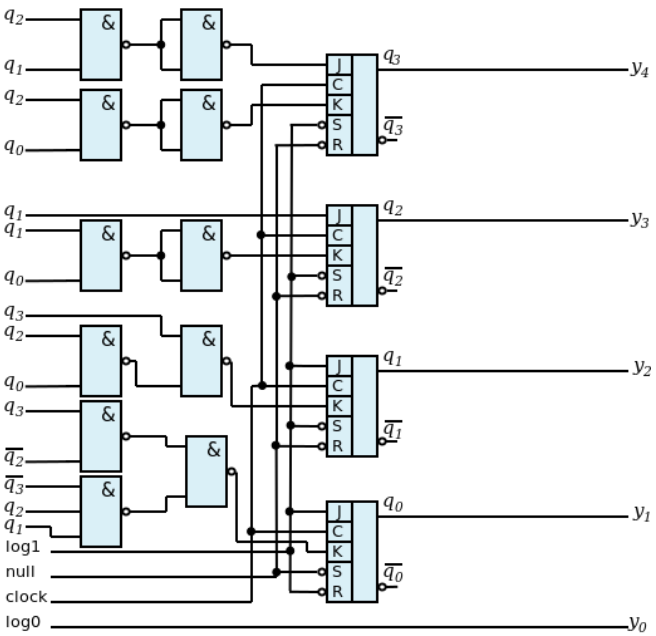
$$y_0 = 0$$

Realizace a nastavení počátečního stavu sekvenčních obvodů D a J-K

V tomto řešení, které nevyužívá dekodéru, je zapotřebí počáteční stav nastavit podle tabulky přechodů do stavu S_1. Je třeba si uvědomit, že proměnná q_0 odpovídá výstupní proměnné y_1, neboť y_0 je trvale připojena k nule.



Obrázek 20: Schéma zapojení generátoru čísel bez dekodéru s obvody D



Obrázek 21: Schéma zapojení generátoru čísel bez dekodéru s obvody J-K

4.1.3 Zhodnocení

Ze schémat jednotlivých způsobů realizací je patrné, že realizace s obvody J-K vyjde o něco levněji z hlediska počtu potřebných hradel. Rovněž minimalizace funkcí a následná kontrola je jednodušší oproti realizaci s obvody D. Zmíněné výhody jsou dány především rozdělením vstupů na dva J a K a přídavnou funkcí invertoru, ve které se obvod nachází v případě, že jsou oba vstupy v úrovni 1.

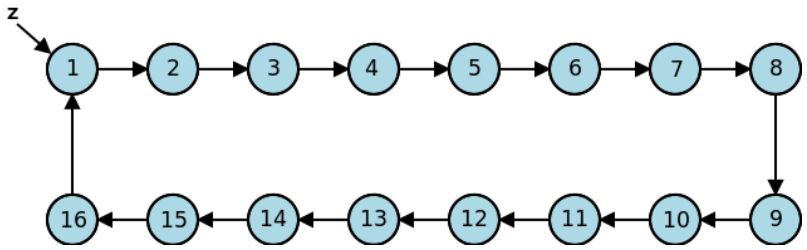
Co se týká otázky využití dekodéru, tak se mi lépe pracovalo s řešeními využívající dekodér, protože to vedlo k rozdělení počtu logických členů do dvou bloků na budící funkce a dekodér. Návrh se tedy jako by rozdělil na dva menší pod úkoly a snáze se s ním pracovalo. Při změně generovaných čísel by se řešila jen část dekodéru.

Při konstrukci generátoru pětibitových čísel bych tedy zvolil postup s využitím obvodů J-K s dekodérem.

4.2 Návrh generátoru binární posloupnosti

Zadaní

V tomto případě se pokusím navrnout generátor 16-bitové posloupnosti. Při návrhu opět využiji obvody D, J-K a nakonec budu návrh demonstrovat s využitím posuvného registru. Zvolená bitová posloupnoust bude ve tvaru: **1010 1100 1110 0001**.



Obrázek 22: Orientovaný graf generátoru 16-bit. poslopnosti s obvody D a J-K

4.2.1 Generátor binární bitové posloupnosti s obvody D a J-K

Stanovení počtu stavových proměnných a zakódování stavů

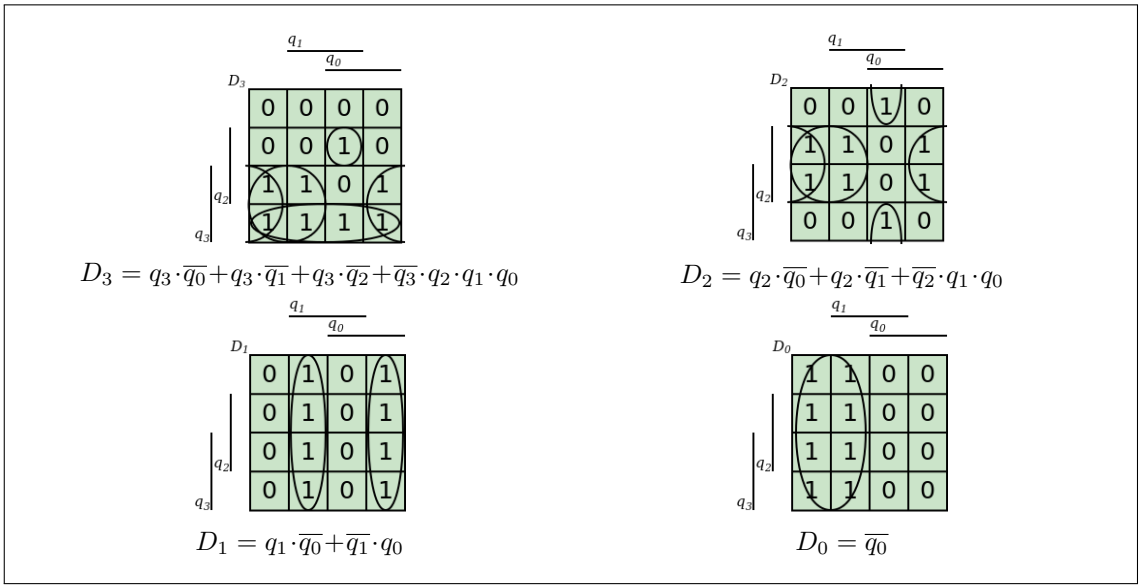
Pro vygenerování sekvence 16 čísel je potřeba stanovit 16 různých stavů, jak vyplývá z orientovaného grafu na obrázku 22. Takovýto počet stavů lze zakódovat pomocí čtyř proměnných q_0 až q_3, jež představují jednotlivé sekvenční obvody, které budou v návrhu použity. Pro generování posloupnosti bitů je třeba za sekvenční obvody zařadit dekodér, který jednotlivé stavy převede do jediné výstupní proměnné y, z jejíhož výstupu lze jednotlivé bity číst.

Zakódování stavů					Následné stavy				Výstupní proměnná
	q_3^t	q_2^t	q_1^t	q_0^t	q_3^{t+1}	q_2^{t+1}	q_1^{t+1}	q_0^{t+1}	y
S_1	0	0	0	0	0	0	0	1	1
S_2	0	0	0	1	0	0	1	0	0
S_3	0	0	1	0	0	0	1	1	1
S_4	0	0	1	1	0	1	0	0	0
S_5	0	1	0	0	0	1	0	1	1
S_6	0	1	0	1	0	1	1	0	1
S_7	0	1	1	0	0	1	1	1	0
S_8	0	1	1	1	1	0	0	0	0
S_9	1	0	0	0	1	0	0	1	1
S_{10}	1	0	0	1	1	0	1	0	1
S_{11}	1	0	1	0	1	0	1	1	1
S_{12}	1	0	1	1	1	1	0	0	0
S_{13}	1	1	0	0	1	1	0	1	0
S_{14}	1	1	0	1	1	1	1	0	0
S_{15}	1	1	1	0	1	1	1	1	0
S_{16}	1	1	1	1	0	0	0	1	1

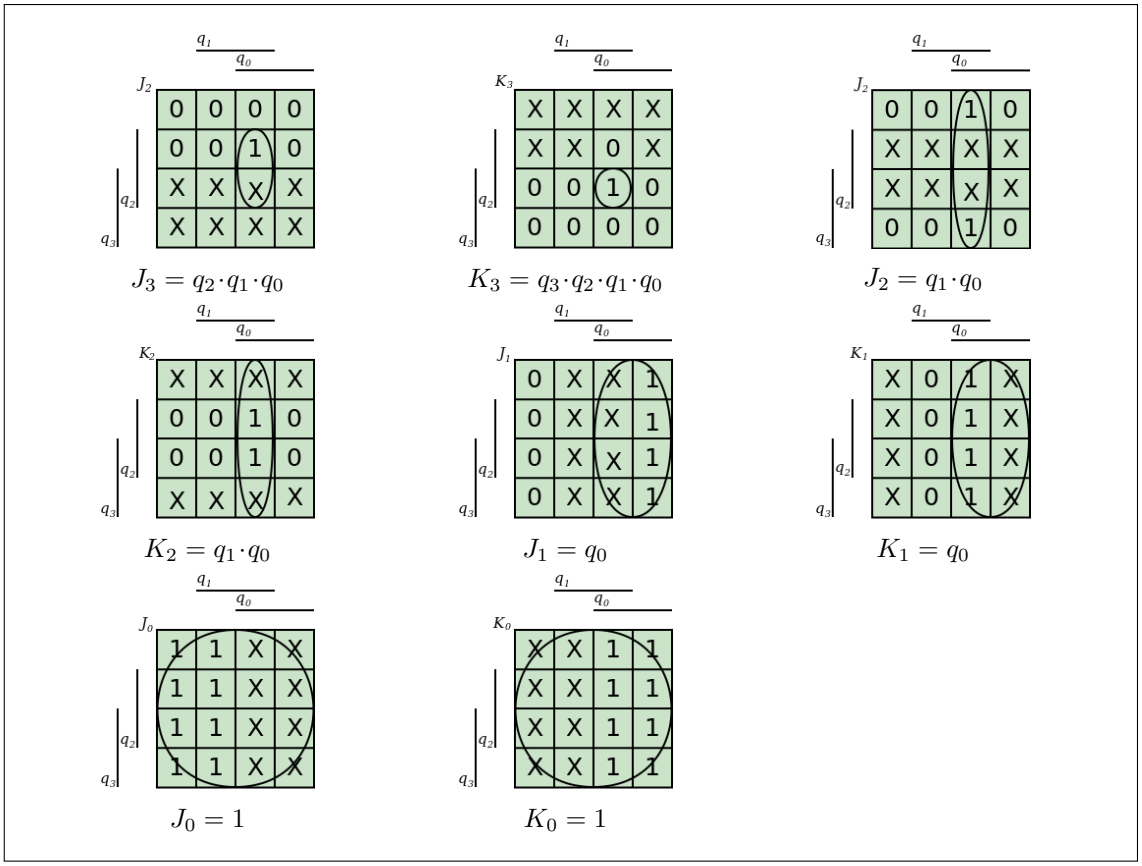
Tabulka 7: Tabulka přechodů pro obě řešení s obvody D i J-K

Sestavení budících Booleovských funkcí pro obvody D a J-K

Pro sestavení Karnaughových map pro budící funkce D a J-K použiji hodnoty z tabulky 7. Budící funkce budou podobné jako u příkladu 4.1.1, ale navíc s proměnnou q_3.



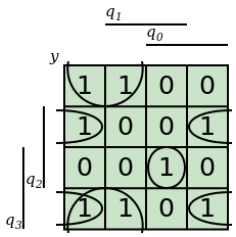
Obrázek 23: Karnaughovy mapy generátoru bit. posloupnosti pro budící funkce D



Obrázek 24: Karnaughovy mapy generátoru bit. posloupnosti pro budící funkce J-K

Sestavení výstupních Booleovských funkcí

Jelikož generátoru bit. posloupnosti má pouze jednu výstupní proměnnou y, je sestavení Booleovských funkcí poměrně jednoduchou záležitostí, jejichž výsledkem bude dekodér, vycházející pouze z jedné Booleovské rovnice.



$$y = \overline{q_2} \cdot \overline{q_0} + \overline{q_3} \cdot q_2 \cdot \overline{q_1} + \overline{q_2} \cdot q_3 \overline{q_1} + q_3 \cdot q_2 \cdot q_1 \cdot q_0$$

Obrázek 25: Karnaughova mapa pro dekodér generátoru bit. posloupnosti

Sestavení rovnic pro realizaci výstupních a budících funkcí s hradly NAND

Výsledné rovnice je opět třeba převést do ekvivalentní formy s hradly NAND. V tomto případě jsou Booleovské rovnice pro obvody D vcelku složité. Komplikovaná je také rovnice pro výstupní funkci y, což ale vyvažuje fakt, že je pouze jedna a to pro obě řešení.

Budící funkce pro řešení s obvody D

$$D_3 = q_3 \cdot \overline{q_0} + q_3 \cdot \overline{q_1} + q_3 \cdot \overline{q_2} + \overline{q_3} \cdot q_2 \cdot q_1 \cdot q_0 = \overline{\overline{q_3 \cdot \overline{q_0} + q_3 \overline{q_1} + q_3 \cdot \overline{q_2} + \overline{q_3} \cdot q_2 \cdot q_1 \cdot q_0}} = \overline{\overline{q_3 \cdot \overline{q_0}} \cdot \overline{q_3 \cdot \overline{q_1}} \cdot \overline{q_3 \cdot \overline{q_2}} \cdot \overline{\overline{q_3} \cdot q_2 \cdot q_1 \cdot q_0}}$$

$$D_2 = q_2 \cdot \overline{q_0} + q_2 \cdot \overline{q_1} + \overline{q_2} \cdot q_1 \cdot q_0 = \overline{q_2 \cdot \overline{q_0} + q_2 \cdot \overline{q_1} + \overline{q_2} \cdot q_1 \cdot q_0} = \overline{q_2 \cdot \overline{q_0}} \cdot \overline{q_2 \cdot \overline{q_1}} \cdot \overline{\overline{q_2} \cdot q_1 \cdot q_0}$$

$$D_1 = q_1 \cdot \overline{q_0} + \overline{q_1} \cdot q_0 = \overline{\overline{q_1 \cdot \overline{q_0}} + \overline{q_1} \cdot q_0} = \overline{\overline{q_1 \cdot \overline{q_0}} \cdot \overline{\overline{q_1} \cdot q_0}}$$

$$D_0 = \overline{q_0}$$

Budící funkce pro řešení s obvody J-K

$$J_3 = q_2 \cdot q_1 \cdot q_0 = \overline{\overline{q_2 \cdot q_1 \cdot q_0}} \qquad K_3 = q_3 \cdot q_2 \cdot q_1 \cdot q_0 = \overline{\overline{q_3 \cdot q_2 \cdot q_1 \cdot q_0}}$$

$$J_2 = q_1 \cdot q_0 = \overline{\overline{q_1 \cdot q_0}} \qquad K_2 = q_1 \cdot q_0 = \overline{\overline{q_1 \cdot q_0}}$$

$$J_1 = q_0 \qquad K_1 = q_0$$

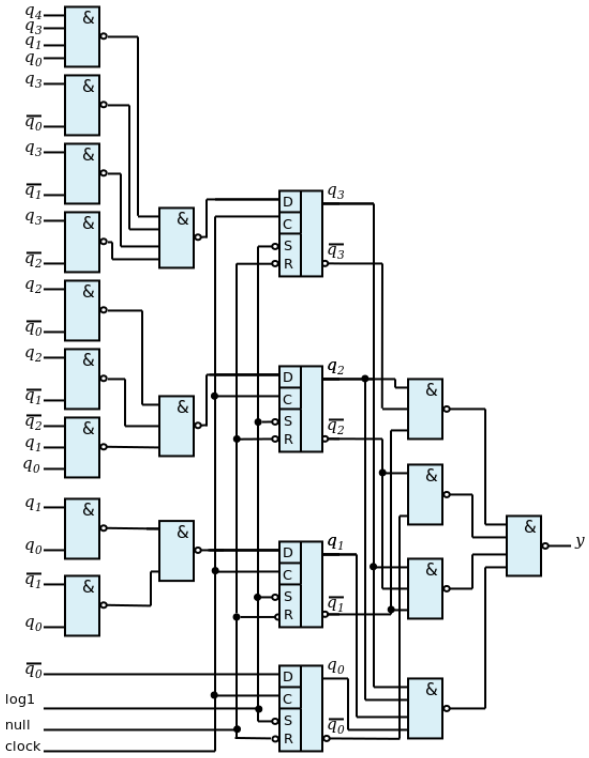
$$J_1 = 1 \qquad K_0 = 1$$

Výstupní funkce y pro obě řešení bez dekodéru

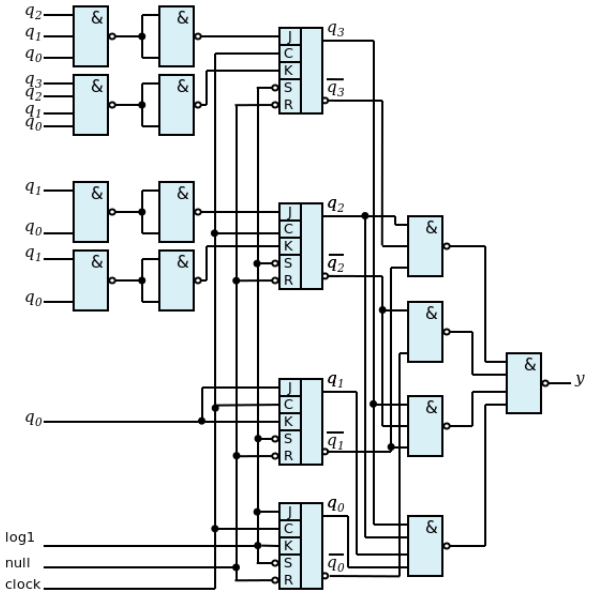
$$y = \overline{q_2} \cdot \overline{q_0} + \overline{q_3} \cdot q_2 \cdot \overline{q_1} + \overline{q_2} \cdot q_3 \cdot \overline{q_1} + q_3 \cdot q_2 \cdot q_1 \cdot q_0 = \overline{\overline{q_2} \cdot \overline{q_0} + \overline{q_3} \cdot q_2 \cdot \overline{q_1} + \overline{q_2} \cdot q_3 \cdot \overline{q_1} + q_3 \cdot q_2 \cdot q_1 \cdot q_0} = \overline{\overline{q_2} \cdot \overline{q_0}} \cdot \overline{\overline{q_3} \cdot q_2 \cdot \overline{q_1}} \cdot \overline{\overline{q_2} \cdot q_3 \cdot \overline{q_1}} \cdot \overline{q_3 \cdot q_2 \cdot q_1 \cdot q_0}$$

Realizace a nastavení počátečního stavu sekvenčních obvodů D a J-K

U obou řešení jak s obvody D, tak s obvody J-K je využit dekodér, což zjednodušuje zapojení logických členu pro budící funkce. Samotný dekodér je sestaven pouze z jedné Booleovské rovnice a jak je patrné z následných dvou schémat, které jsou už tak dost složité, jeho nezařazení by výsledné schéma hodně zkomplikovalo.



Obrázek 26: Schéma zapojení generátoru bit. posloupnosti s obvody D

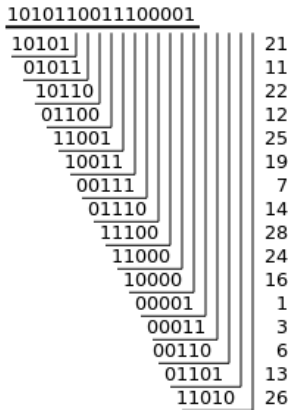


Obrázek 27: Schéma zapojení generátoru bit. posloupnosti s obvody J-K

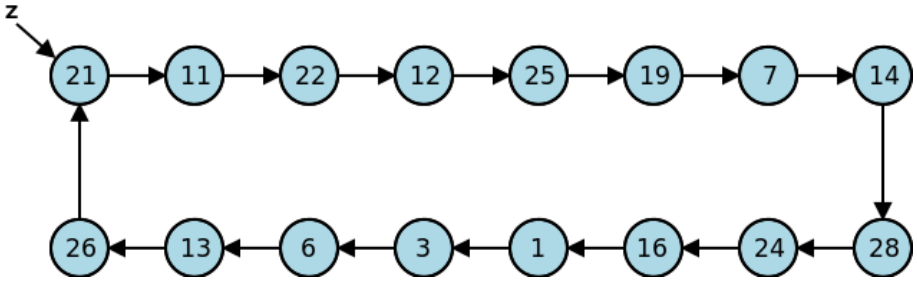
4.2.2 Generátor binární posloupnosti s posuvným registrem

Stanovení počtu stavových proměnných a zakódování stavů

Pro sestavení generátoru 16 bitové binární posloupnosti s posuvným registrem bych si měl teoreticky vystačit se čtyřmi obvody D podle vzorce $d \leq 2^n$, kde d je počet stavů a n je počet sekvenčních obvodů. Vzhledem k tomu, že by transformace do orientovaného grafu nebyla jednoznačná, stav 12 by se opakoval dvakrát, stanovím $n = 5$. Tím se zvýší počet obvodů D na pět a zároveň počet možných stavů na 32. S tímto počtem již bez problému generátor sestavím. Zároveň se musí provést patřičná úprava orientovaného grafu z obrázku 22.



Obrázek 28: Rozdělení binární posloupnosti do pětibitových čísel



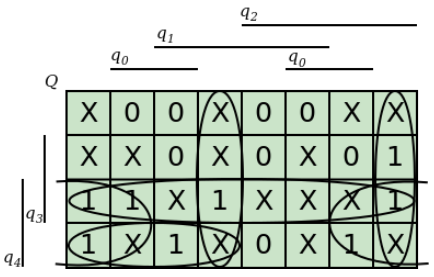
Obrázek 29: Orientovaný graf pro binární posloupnost n = 5

Sestavení budících Booleovských funkcí posuvného registru

Vzhledem k tomu, že budící funkcí Q generuji jednotlivé stavy vstupující na vstup obvodu D_0, stačí k sestavení celé činnosti jediná funkce, kterou odvodím z Karnaughovy mapy na Obrázku 30. Budící funkce sériového vstupu je poměrně komplikovaná, ale je pouze jedna, a proto bude potřeba jen jedna Karnaughova mapa pro její minimalizaci.

Zakódování stavů						Budící funkce
	q_4	q_3	q_2	q_1	q_0	Q
S_{21}	1	0	1	0	1	1
S_{11}	0	1	0	1	1	0
S_{22}	1	0	1	1	0	0
S_{12}	0	1	1	0	0	1
S_{25}	1	1	0	0	1	1
S_{19}	1	0	0	1	1	1
S_7	0	0	1	1	1	0
S_{14}	0	1	1	1	0	0
S_{28}	1	1	1	0	0	0
S_{24}	1	1	0	0	0	0
S_{16}	1	0	0	0	0	1
S_1	0	0	0	0	1	1
S_3	0	0	0	1	1	0
S_6	0	0	1	1	0	1
S_{13}	0	1	1	0	1	0
S_{26}	1	1	0	1	0	1

Tabulka 8: Pravdivostní tabulka budící funkce Q sériového vstupu



$$Q = \overline{q_4}\cdot\overline{q_1}\cdot\overline{q_0}+\overline{q_4}\cdot\overline{q_2}\cdot\overline{q_1}+\overline{q_2}\cdot\overline{q_1}\cdot q_0+q_4\cdot\overline{q_3}\cdot\overline{q_2}+\overline{q_2}\cdot q_1\cdot\overline{q_0}+\overline{q_4}\cdot\overline{q_3}\cdot q_1\cdot\overline{q_0}+q_4\cdot q_2\cdot q_0$$

Obrázek 30: Karnaughova mapa pro budící funkci Q sériového vstupu

Sestavení rovnic pro realizaci budících a výstupních funkcí s hradly NAND

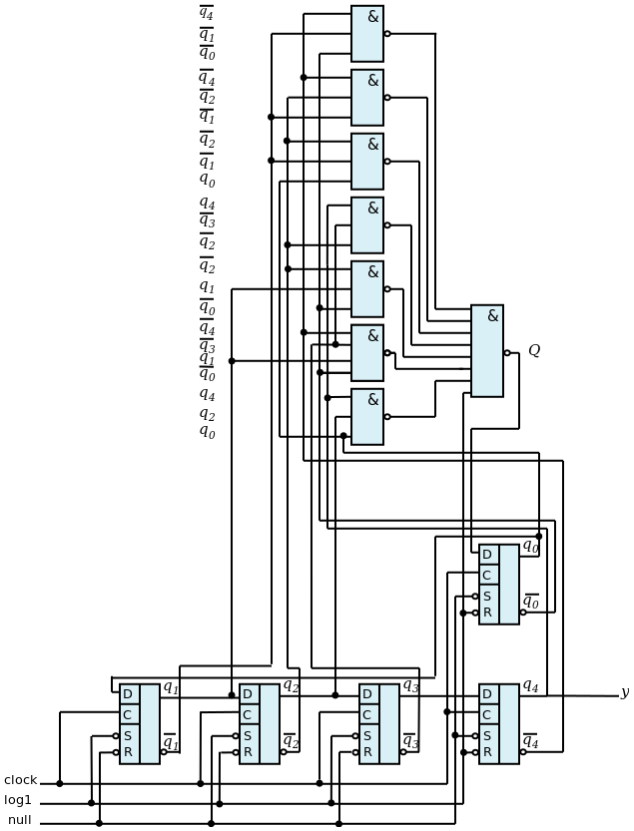
Z předešlého postupu vyplývá, že převod na ekvivalentní formu nebude nikterak složitý. Převede se pouze booleovská funkce sériového vstupu, jejímž výsledkem bude osm logických členů NAND zapojených tak, aby se na vstupu obvodu D_0 generovala zadaná posloupnost.

Budící funkce sériového vstupu Q

$$Q = \overline{q_4}\cdot\overline{q_1}\cdot\overline{q_0}\cdot+\overline{q_4}\cdot\overline{q_2}\cdot\overline{q_1}+\overline{q_2}\cdot\overline{q_1}\cdot q_0+q_4\cdot\overline{q_3}\cdot\overline{q_2}+\overline{q_2}\cdot q_1\cdot\overline{q_0}+\overline{q_4}\cdot\overline{q_3}\cdot q_1\cdot\overline{q_0}+q_4\cdot q_2\cdot q_0 =$$
$$\overline{\overline{q_4}\cdot\overline{q_1}\cdot\overline{q_0}+\overline{q_4}\cdot\overline{q_2}\cdot\overline{q_1}+\overline{q_2}\cdot\overline{q_1}\cdot q_0+q_4\cdot\overline{q_3}\cdot\overline{q_2}+\overline{q_2}\cdot q_1\cdot\overline{q_0}+\overline{q_4}\cdot\overline{q_3}\cdot q_1\cdot\overline{q_0}+q_4\cdot q_2\cdot q_0} =$$
$$\overline{\overline{q_4}\cdot\overline{q_1}\cdot\overline{q_0}\cdot\overline{q_4}\cdot\overline{q_2}\cdot\overline{q_1}\cdot\overline{q_2}\cdot\overline{q_1}\cdot q_0\cdot\overline{q_4}\cdot\overline{q_3}\cdot\overline{q_2}\cdot\overline{q_2}\cdot q_1\cdot\overline{q_0}\cdot\overline{q_4}\cdot\overline{q_3}\cdot q_1\overline{q_0}\cdot\overline{q_4}\cdot q_2\cdot q_0}$$

Realizace a nastavení počátečního stavu sekvenčních obvodů a registru

Při realizaci generátoru 16-bit. posloupnosti z posuvným registrem využiji méně logických členů NAND, než v předchozích dvou příkladech, jak lze vidět na obrázku 31.Výhodou je také potřeba pouze jedné budící funkce, která se přiřadí na vstup D_0.



Obrázek 31: Realizace zapojení generátoru 16-bit. posloupnosti s posuvným registrem

4.2.3 Porovnání mezi řešení s registrem a s obvody D a J-K

Ze schémat je opět patrné, že k realizaci generátoru bitové posloupnosti s obvody D je potřeba větší množství logických členů oproti realizaci s obvody J-K. Při použití posuvného registru se množství logických členů NAND, které bylo potřeba použít, snížil

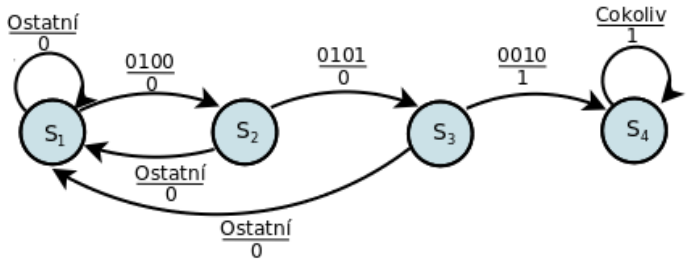
zhruba na polovinu než u zapojení s obvody J-K. To je dáno zejména tím, že odpadá nutnost zařadit dekodér.

Pro realizaci generátoru 16 bitové posloupnosti posuvného registru SISO. bych tedy volil postup s využitím obvodů J-K, nebo posuvného registru.

4.3 Elektronický zámek s trojcifernou kombinací čísel 0 až 9

Zadaní

V tomto posledním příkladu se pokusím navrhnout elektronický zámek s trojcifernou kombinací čísel 0 až 9. Toto zapojení se bude od ostatních lišit a to zejména v tom, že bude přijímat zadaná čísla v podobě vstupních proměnných. Po zadání těchto čísel ve správné posloupnosti by se měl zámek otevřít. Za povšimnutí také stojí, že tento návrh je Mealyho typu, zatímco oba předešlé příklady jsou typu Moorova.



Obrázek 32: Orientovaný graf el.zámku

Stanovení počtu stavových proměnných a zakódování stavů

Činnost trojciferného zámku lze popsat pomocí orientovaného grafu32. Stavy zakóduji pomocí dvou proměnných q_0 a q_1. Pro zadávání čísel 0 až 9 bude potřeba čtyřech vstupních proměnných x_0 až x_3. Zvolím si také správnou kombinaci zadaných čísel 462. Návrh jsem se rozhodl realizovat pomocí obvodů D.

Tabulka 9 odráží stav výstupní proměnné y v každém stavu. Z orientovaného grafu 32 lze také vyčíst, že k dosáhnutí konečného stavu S_4 musí být zároveň splněna podmínka průchodu předcházejícími stavy (zadáni správné posloupnosti čísel).

Funkci orientovaného grafu 32 lze popsat následovně:

Stav S_1 je výchozí bod, který očekává číslici v rozsahu 0 až 9. Po zadání hodnoty 4 se automat posune do stavu S_2. Po zadání číslice 6 se automat posune do stavu S_3. Po zadaní třetí číslice 2 se automat dostane do konečného stavu S_4, kde již nereaguje na žádné další hodnoty. Jestliže je ve stavech S_1 až S_3 zadaná nesprávná číslice dostává se automat do výchozího stavu S_1.

Stav	Vnitřní proměnné		Výstupní proměnná
	q_1	q_0	y
S_1	0	0	0
S_2	0	1	0
S_3	1	0	0
S_4	1	1	1

Tabulka 9: Zakódování stavů a výstupní proměnná

Stanovení následných stavů pro každý stav obvodu a každou kombinaci vstupních proměnných

Tabulka přechodů 10 je odlišná od předchozích tabulek, protože musí brát ohled na stavy vstupních proměnných x v každém stavu S. V každém sloupci tabulky jsou dvě hodnoty, první pro proměnou q_1, druhá pak pro proměnnou q_0. Proměnné q představují jednotlivé obvody D v realizaci el. zámku.

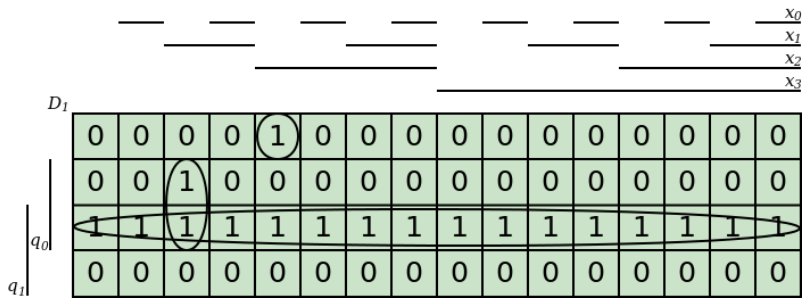
Stav	Následné stavy: q_i^{t+1} Hodnoty vstupních proměnných: $x_4x_3x_2x_1$															
$q_1^t q_0^t$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
S_1	0 0	0 0	0 0	0 0	0 1	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
S_2	0 0	0 0	0 0	0 0	0 0	0 0	1 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
S_3	0 0	0 0	1 1	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
S_4	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1

Tabulka 10: Tabulka přechodů el. zámku

Sestavení budících Booleovských funkcí s obvody D

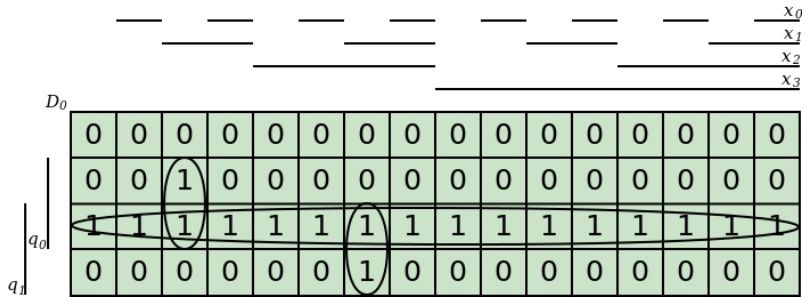
Karnaughovy mapy sestavené z tabulky 10 vypadají na první pohled složitě a nepřehledně. Při bližším prozkoumání si lze však všimnout, že každý sloupec mapy představuje hodnotu zadávanou do elektrického zámku od čísla 0 až po číslo 15.

Přestože použiji Karnaughovy mapy pro 6 proměnných, díky malému množství logických 1, které odpovídají správně zadanému číslu, jsou poměrně přehledné.



$$D_1 = \overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0} \cdot q_0 + \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot \overline{x_0} \cdot \overline{q_1} \cdot \overline{q_0} + q_1 \cdot q_0$$

Obrázek 33: Karnaughova mapa pro budící funkci D_1

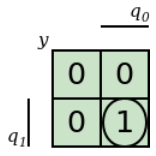


$$D_0 = \overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0} \cdot q_0 + \overline{x_3} \cdot x_2 \cdot x_1 \cdot \overline{x_0} \cdot q_1 + q_1 \cdot q_0$$

Obrázek 34: Karnaughova mapa pro budící funkci D_0

Sestavení výstupních Booleovských funkcí

Karnaughova mapa pro výstupní funkci y je velice jednoduchá, protože zahrnuje pouze dvě proměnné q_1 a q_0. Jestliže jsou obě tyto proměnné v logické 1, je zámek otevřen.



$$y = q_1 \cdot q_0$$

Obrázek 35: Karnaughova mapa pro výstupní funkci y

Sestavení rovnic pro realizaci budících a výstupních funkcí s hradly NAND

Při sestavování ekvivalentních rovnic vyjdu opět z minimální Booleovské formy, odvozené z výše uvedených Karnaughových map.

Budící funkce jednotlivých obvodů D

$$D_1 = \overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot\overline{x_0}\cdot q_0 + \overline{x_3}\cdot x_2\cdot\overline{x_1}\cdot\overline{x_0}\cdot\overline{q_1}\cdot\overline{q_0} + q_1\cdot q_0 =$$
$$\overline{\overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot\overline{x_0}\cdot q_0 + \overline{x_3}\cdot x_2\cdot\overline{x_1}\cdot\overline{x_0}\cdot\overline{q_1}\cdot\overline{q_0} + q_1\cdot q_0} =$$
$$\overline{\overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot\overline{x_0}\cdot q_0}\cdot\overline{\overline{x_3}\cdot x_2\overline{x_1}\cdot\overline{x_0}\cdot\overline{q_1}\cdot\overline{q_0}}\cdot\overline{q_1\cdot q_0}$$

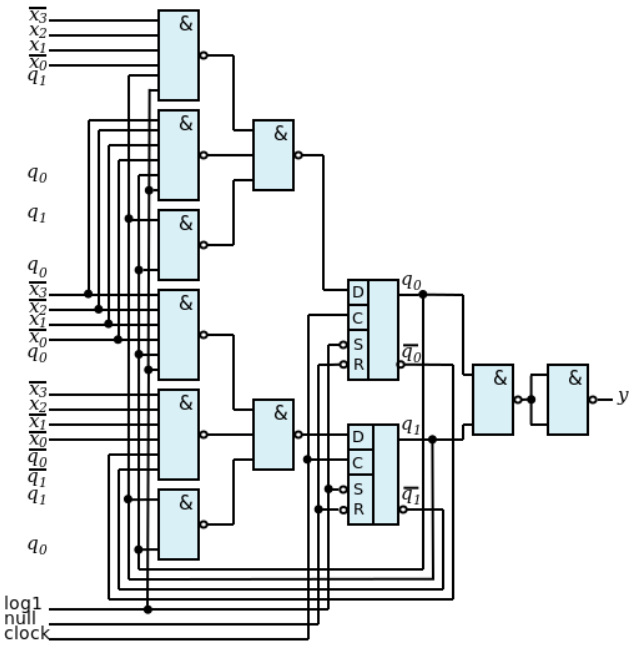
$$D_0 = \overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot\overline{x_0}\cdot q_0 + \overline{x_3}\cdot x_2\cdot x_1\cdot\overline{x_0}\cdot q_1 + q_1\cdot q_0 =$$
$$\overline{\overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot\overline{x_0}\cdot q_0 + \overline{x_3}\cdot x_2\cdot x_1\cdot\overline{x_0}\cdot q_1 + q_1\cdot q_0} =$$
$$\overline{\overline{x_3}\cdot\overline{x_2}\cdot x_1\cdot\overline{x_0}\cdot q_0}\cdot\overline{\overline{x_3}\cdot x_2\cdot x_1\cdot\overline{x_0}\cdot q_1}\cdot\overline{q_1\cdot q_0}$$

Výstupní proměnná y

$y = q_1\cdot q_2 = \overline{\overline{q_1\cdot q_2}}$

Realizace a nastavení počátečního stavu elektronického zámku

Výsledné schéma je proti předchozím příkladům relativně jednoduché, ale je nutno dodat, že v něm nejsou zahrnuty invertory pro negování vstupních proměnných x.



Obrázek 36: Realizace elektronického zámku

Zhodnocení

Ze schématu jde vidět, že realizovat elektronický trojciferný zámek je relativně snadné v porovnání s předchozími příklady a to i navzdory rozsáhlosti Karnaughových map pro budící funkce. K uchování troj-číselné kombinace jsem si vystačil pouze s dvojicí

sekvenčních obvodů D. Oproti předchozím příkladům vyvstala potřeba zahrnutí čtyř vstupních proměnných x_0 až x_3 do Karnaughových map. Proměnné q_0 až $q1$ slouží k uchování předešlých stavů zámku a zajišťují to, aby byla čísla zadána ve správném pořadí. Návrh zámku také nepočítá s resetováním do výchozího stavu, například pro tento účel zvoleným tlačítkem.

5 Java a knihovna Swing

Pro vytvoření výstupu v podobě počítačového programu, který simuluje jednotlivá zapojení, jsem si zvolil programovací jazyk Java. K vytvoření jednotlivých komponent, které se v programu vyskytují jsem využil knihovny Swing[3][4], která je sice primárně učena pro tvorbu uživatelského rozhraní (GUI), ale lze s ní vytvořit plynulé animace[5] a také vcelku jednoduše vykreslovat jednoduché tvary jako jsou např. obdelník ovál a další. Pomocí těchto jednoduchých primitivů lze skládat složitější obrazce, jako třeba mnou vytvořené logické členy NAND.

Jazyk Java byl poprvé představen firmou Sun Microsystems v roce 1995. Java není pouze programovací jazyk, ale spíše platforma, nad kterou mohou běžet programy napsané v Javě. Programy vykonává tzv. "Javoská mašina"JVM, která se stará o celý runtime. Nejprve vezme zdrojové kódy a přeloží je do bytecódu, který se podobá Assembleru a takto přeložený kód uloží do souborů s připonou .class. Takto přeložené soubory je JVM schopná číst a vykonávat instrukci za instrukcí. JVM má na starosti také úklid nepotřebných objektů, ke kterému používá Garbage collector. Toto je velká výhoda pro programátory, protože se mohou soustředit na vývoj místo hlídání úniků paměti jako u jazyka C++.

Nespornou výhodou je také fakt, že je Java nezávislá na platformě. To znamená, že programy napsané v Javě můžou běžet jak na systémech Windows, tak na systémech Linux nebo počítačích od firmy Apple. Společnost Sun proto s Javou spojuje heslo "Write once, run anywhere".

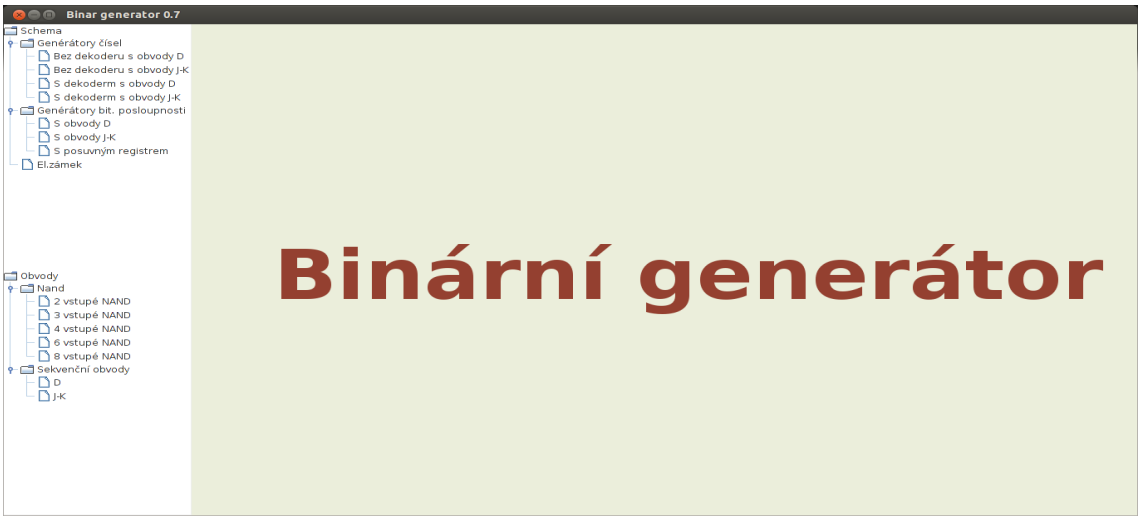
6 Binární generátor

Program Binární generátor simuluje všechny předešlé návrhy se sekvenčními obvody. V této kapitole se budu snažit popsat, jak se program ovládá a na jakém principu funguje.

6.1 Možnosti Binárního generátoru

Jak jsem již uvedl v předchozí kapitole, je program Binární generátor napsán v Javě. Pro jeho spuštění je tedy potřeba, aby bylo na konkrétním PC nainstalováno JDK ve verzi 7 a výš. Program by měl fungovat na jakémkoliv operačním systému, testován byl na Linuxu i Windows.

Program se spouští jednoduše poklepáním na jar soubor Binarni generator. Po spuštění se objeví úvodní obrazovka s titulním nápisem Binární generátor. Po levé straně je ve stromové struktuře nabídka výběru jednotlivých zapojení. Pod nabídkou zapojení je další menu s výběrem jednotlivých obvodů, které lze do jednotlivých zapojení přidávat.



Obrázek 37: Úvodni obrazovka Binárního generátoru

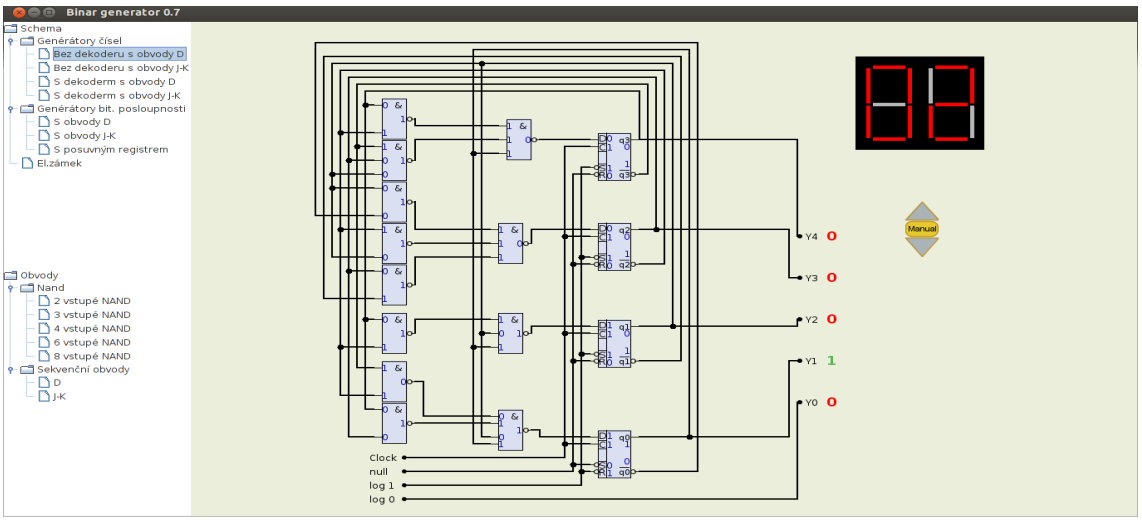
Menu, kde se vybírají jednotlivá zapojení je v kořenovém adresáři Schema, které je rozděleno do tří logických celků a to: Generátory čísel, Generátory bit. posloupnosti a posledním, pouze s jedním zapojením, s názvem El. Zámek.

Druhé menu pro výběr jednotlivých obvodů (Obvody) má dvě části. První oddíl, s názvem Nand, pro výběr jednotlivých logických členů a druhý, jehož jméno je Sekvenční obvody, pro výběr obvodů D a J-K.

V následujících třech podkapitolách popíší princip činnosti jednotlivých celků.

6.1.1 Generátory čísel

Některý ze čtyř generátorů čísel lze vybrat tak, že dvojklikem poklepeme na příslušné zapojení. Posléze se objeví vybrané zapojení v podobě interaktivního el. schématu uvedených v kapitole 4.1. Kromě logických obvodů a spojnic, které představují propojení mezi vstupy a výstupy jednotlivých komponent, je ve schématu také vidět displej a klikací ovladač pro nastavení rychlosti hodinového signálu Clk.



Obrázek 38: Generátor čísel s obvody D bez dekodéru

Klikací ovladač, který se skládá ze tří částí, může být přepnut do režimu manual, nebo auto tím, že se myší klepne do jeho prostřední části v podobě zaobleného obdelníku. V manualním režimu se kliknutím na kteroukoliv ze šipek ovladače invertuje stav na výstupu Clk. V automatickém režimu se invertuje signál výstupu Clk na základě rychlosti zobrazené v prostřední části ovladače. Rychlost, kterou lze nastavit v rozmezí 1 až 8, se určuje šipkami tak, že klepnutím na vrchní šipku se rychlost sníží a při klepnutí na dolní šipku zase sníží.

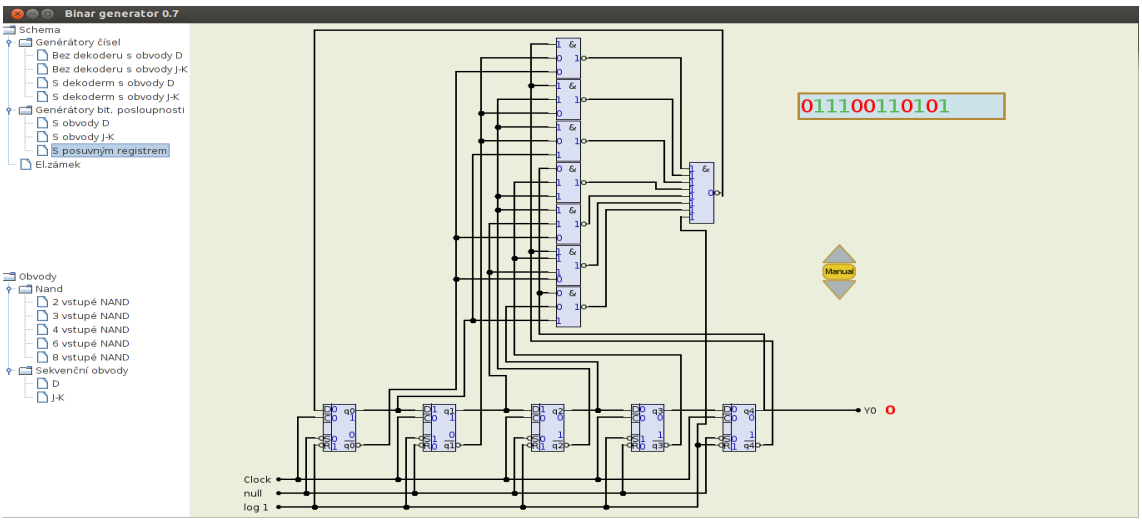
Stavy, ve kterých se generátor nachází, lze číst jak z jednotlivých výstupů Y v binární podobě, tak jako dekadické číslo z displaye , který lze na obrázku 38 také vidět. Displej napodobuje sedmi-segmentový led displej, používaný v elektronice pro zobrazování číselných údajů.

Všechny komponenty, které se ve všech schématech objevují, lze libovolně přemisťovat. Obvody a jednotlivé spoje mohou být navíc do schématu přidávány nebo mazány. Kreslení spojnic se provádí stiskem levého tlačítka myši a následného přesunuti myši na zamýšlenou pozici. Následným uvolněním levého tlačítka se přidá spojnice do schématu. Mazáni jak obvodů, tak spojnic se docílí pomocí pravého tlačítka myši.

6.1.2 Generátory bitové posloupnosti

Další skupinou, kterou lze vybrat jsou tři schémata Generátorů bit. posloupnosti. Po dvojitém kliknutí na některé z nich se objeví některé z řešení kapitoly 4.2. Jako v předchozím případě jsou všechna schémata interaktivní a obsahují prvky k jejich ovládání.

Každé schéma obsahuje opět logické členy a spojnice sestavené tak, aby simulovaly jednotlivá řešení. Každé řešení obsahuje také ovladač rychlosti, který funguje stejným způsobem jako u generátorů čísel. Místo displeje však schémata obsahují registr pro ukládání bitové posloupnosti, jak lze vidět na obrázku 39. Po naplnění registru 16 hodnotami se registr vyprázdní a generování posloupnosti začíná od začátku.

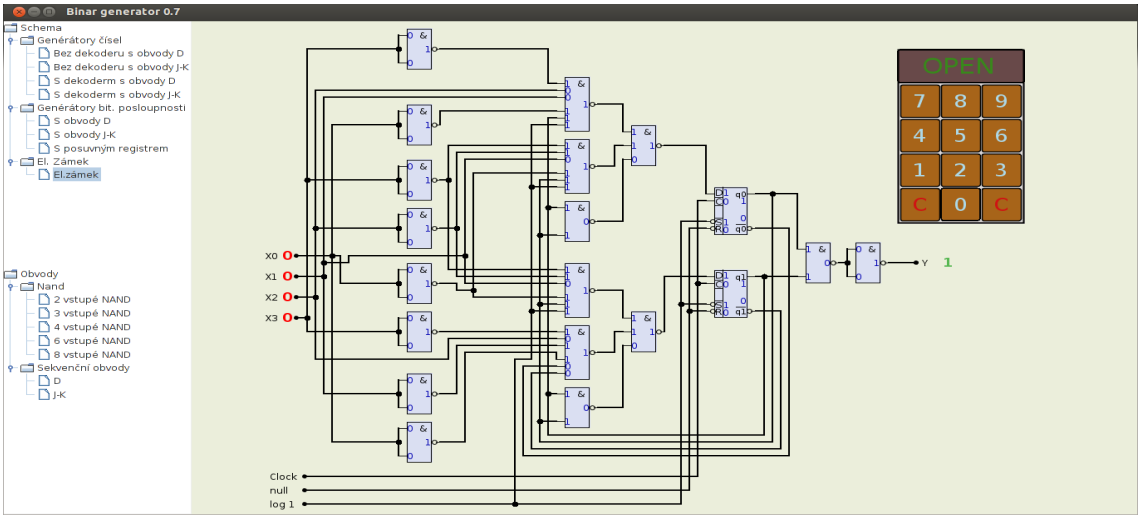


Obrázek 39: Generátor bitové posloupnosti v provedení s posuvným regisrem

6.1.3 Elektronický trojciferny zámek

Jako třetí v pořadí, lze vybrat skupinu Elektronický zámek, který obsahuje pouze jedno interaktivní schéma se shodným názvem. Elektronický zámek má klávesnici pro zadávaní číselných kombinací a neobsahuje ovladač pro nastavení rychlosti hodinového signálu clk. protože výstup clk. ovládá stisknutí některé z klávesnic a to tak, že pokud je klávesnice stisknuta na výstupu clk. se objeví logická 1.

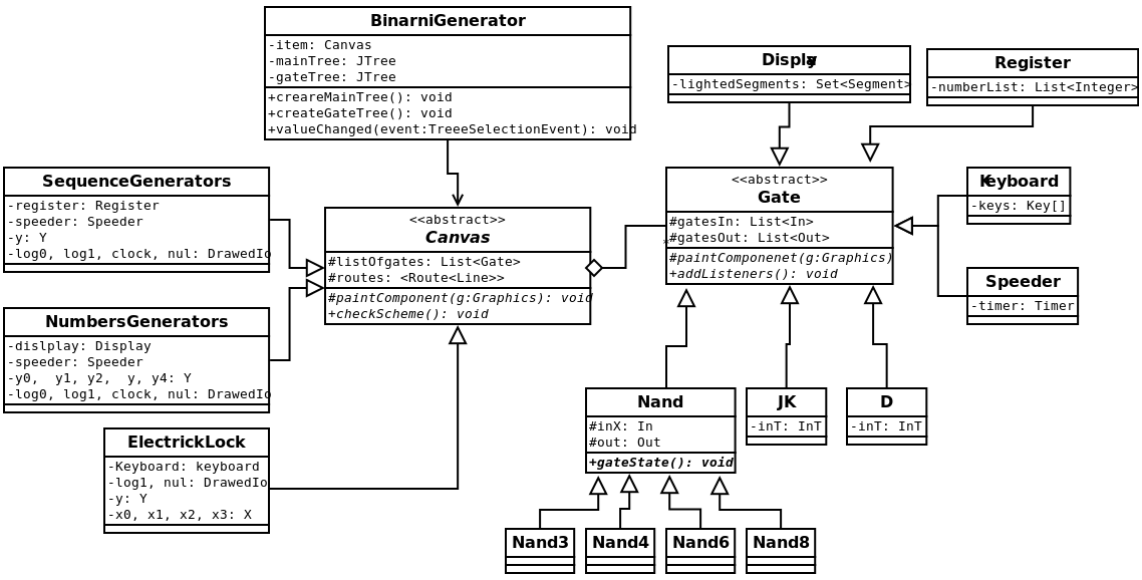
Elektronický zámek funguje podle návrhu z kapitoly 4.3. Otvírá se pouze po zadání správné kombinace čísel 462. Pokud je číselná kombinace zadána nesprávně nebo v odlišném pořadí, zámek se dostane do výchozího stavu. Po zadání správné kombinace čísel, změní zámek stav do otevřeného stavu, který je signalizován nápisem open v horní části klávesnice a výstupní proměnnou Y, jak lze vyčíst z obrázku 40. V tomto stavu již zámek nereaguje na stisk žádné číslice a musí se vyresetovat stiskem některého z tlačítek C.



Obrázek 40: Elektronický trojciferný zámek

6.2 Princip Binárního generátoru

Program Binární generátor se skládá z mnoha tříd a tisíce řádků, a proto vysvětlím jeho princip ve zjednodušené formě. Od toho se odvíjí i výsledný UML diagram41, který zahrnuje jen nejdůležitější třídy a metody nezbytné ke smysluplnému vysvětlení principu programu.



Obrázek 41: Zjednodušený UML diagram Binárního generátoru

Hlavní třídou, která má na starosti spuštění programu a sestavení menu je třída BinarGenerator. Jak lze vyčíst z ULM diagramu, volby výběru jsou utvořeny pomocí objektů JTree a plátna (Canvas), na které se kreslí jednotlivá zapojení. Každé schéma (Canvas) je uloženo v proměnné Item.

Dále si lze všimnout, že abstraktní třída Canvas zajišťuje veškerou funkčnost pro jednotlivá zapojení, která jsou z ní odvozena a reprezentovány Třídami SequenceGenerators, NumbersGenertors a třídou ElectrickLock.

Canvas obsahuje metodu paintComponent, kterou může překreslit celé schéma a také metodu checkScheme, pomocí které je schopen zkontrolovat stav všech obvodů ve schématu. Dále uchovává odkazy kolekcí všech objektů a spojnic z názvy listOfGates a routes. Pomocí těchto kolekcí přistupuje k jednotlivým objektům, které pak může ovládat k tomu určenými metodami.

Samotná třída Gate je pouze abstraktní obálkou, která zaručuje všem z ní odvozeným třídám funkce jako jsou přesouvání a schopnost se nakreslit (addListener, paintComponent). Uchovává také kolekce vstupů a výstupů, které si jednotliví potomci můžou plnit potřebným počtem těchto objektů.

V diagramu jsou dále vidět objekty jako jsou třídy Nand, Display a další, které vlastnosti třídy Gate dědí a rozšiřují o další vlastnosti jim typické. Za vzhled těchto tříd odpovídá metoda paintComponent, ve které je zapsáno jak se má vykreslit celý objekt na obrazovku.

```
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g.create();

    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON);

    g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
            alpha));

    int x = gatePositionX - lastGatePositionX;
    int y = gatePositionY - lastGatePositionY;

    g2.setClip(x, y, width, height);
    g2.setColor(Color.black);
    g2.drawLine(x, inA.position.y + y, ioLength + x, inA.position.y + y);
    g2.drawLine(x, y + inB.position.y, x + ioLength, y + inB.position.y);
    g2.drawLine(width - 5 + x, out.position.y + y, width + x,
            out.position.y + y);
    g2.drawArc(x + width - 10, y + out.position.y -3, 6, 6, 0, 360);

    g2.drawString("&", 25 + x, 15 + y);

    setIoStates(x, y, g2);

    g2.dispose();
}
```

Obrázek 42: Implementace metody paintComponent

Závěr

Na začátku této práce jsem si kladl za cíl popsat jednotlivé typy sekvenčních obvodů a jejich využití, což jsem udělal s přihlédnutím na typy obvodů, které jsem použil v mé závěrečné práci. Popsal jsem také stručně historii a vývoj sekvenčních logických obvodů. Více jsem se rozepsal o posuvných registrech, kde si myslím, že jejich význam ve výpočetní technice je mnohem větší než u jednoduchých sekvenčních obvodů. Zapojení s registrem jsem proto využil v jednom ze zapojení, abych zjistil jak ovlivní celý návrh oproti návrhu s obvody D a J-K.

Samotný návrh se sekvenčními obvody se mi zdál zpočátku vcelku složitý a to zejména proto, že předmět Logické obvody byl v mém ročníku upraven tak, aby více vyhovoval potřebám informatiků, a proto jsem se s těmito obvody setkal poprvé při psaní této práce.

Jak jsem postupoval v psaní této práce a to zejména v kapitole 4, uvědomil jsem si, jak je takový návrh pracný. Nejvíce to bylo poznat při sestavování Karnaughovy mapy pro posuvný registr 30, kde se v mapě o velikosti 64 polí dá udělat velice snadno chyba a ještě hůře se hledá. Snad i proto se v dnešní době podobná zapojení realizují pomocí mikrokontroléru, které mají ve své paměti zapsán k tomu určený program, nejčastěji v jazyce C.

Co se týče praktické části mé práce, byla pro mě situace o něco méně komplikovaná. Nejen proto, že se jednalo o programování v jazyce Java, kterému přece jenom o něco více rozumím. Při psaní jednotlivých komponent, myslím tím obvody D, JK a např. displej, jsem se snažil aby vypadaly vzhledově co nejvíce přesně k jejich vzorům. Vlastně jsem si připadal jako kdybych skládal stavebnicí a postupně do ní přidával a vymýšlel nové díly.

Nakonec jsem tedy splnil všechny cíle, které jsem si na začátku práce předsevzal. Výsledný program lze použít při vysvětlení základních principu činnosti sekvenčních obvodů např. v kurzu Logické obvody. Celý program, který nabízí celou řadu funkcí, by šlo dále ještě vylepšit a nabídnout další užitečné funkce a nakonec z něj udělat plnohodnotný interaktivní CAD program.

7 Reference

[1] ANTOŠOVÁ, Marcela a Vratislav DAVÍDEK. Číslicová technika. 4. aktualiz. vyd. České Budějovice: Kopp, 2009, 305 s. ISBN 978-80-7232-394-4.

[2] MALINA, Václav.Poznáváme elektroniku. 1. vyd. České Budějovice: Kopp, 2006. ISBN 80-7232-271-0.

[3] GUTZ, Steven J. Up to speed with Swing: user interfaces with Java foundation classes. 2nd ed. Greenwich, CT: Manning, c2000, xxxvi, 536 p. ISBN 18-847-7775-9. +

[4] LOY, Marc, Robert ECKSTEIN a Prashant SRIDHARAN. Java Swing. 2nd ed. Sebastopol, CA: O'Reilly, c2003, xxiv, 1252 p. ISBN 05-960-0408-7.

[5] HAASE, Chet a Romain GUY. Filthy rich clients: developing animated and graphical effects for desktop Java applications. Upper Saddle River: Prentice Hall, 2008, xxvii, 572 s. the Java series. ISBN 978-0-13-241393-0.

[6] Http://mikrokontrolery-pic.cz/zaciname/cislicova-technika/sekvencni-logicke-obvody/. [online]. [cit.013-11-23].Dostupné z: Http://mikrokontrolery-pic.cz/zaciname/cislicova-technika/sekvencni-logicke-obvody/

[7] Http://www.root.cz/clanky/od-logickych-obvodu-k-mikroprocesorum/. In: [online]. [cit. 2013-11-23]. Dostupné z: http://www.root.cz/clanky/od-logickych-obvodu-k-mikroprocesorum/

[8] V6 English Tutorial (PDF). [online]. [cit. 2013-11-25]. Dostupné z: http://www.cadsoftusa.com/training/tutorials/?language=en

[9] DIVIŠ, Zdeněk, Zdeňka CHMELÍKOVÁ a Jaroslav ZDRÁLEK. Logické obvody. 1. vyd. s-3.

[10] PETŘÍKOVÁ, Iva, Zdeňka CHMELÍKOVÁ a Jaroslav ZDRÁLEK. Logické obvody: příklady. 1.vyd. Ostrava: Vysoká škola báňská - Technická univerzita, 2001, 155 s. ISBN 80-707-8925-5.